Attorney's Docket No. 030681-152

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	
Euee-seon Jang et al.	Group Art Unit: 2613
Application No.: 09/396,470	Examiner: B. Senfi
Filed: September 15, 1999	Appeal No.: 7479
For: METHOD OF PROGRESSIVELY CODING/DECODING 3-D MESH INFORMATION AND APPARATUS THEREOF	

BRIEF OF THE APPELLANT

Mail Stop APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This appeal is from the decision of the Primary Examiner dated January 3, 2005 (Paper No. 20041219), finally rejecting claims 1-6, 10, 12-15 and 19, which are reproduced as the Claims Appendix A of this brief.

☐ A check covering the ☐ $250.00 (2402) ☒ $500.00 (1402) Government fee is filed herewith.

☐ Charge ☐ $250.00 (2402) ☐ $500.00 (1402) to Credit Card. Form PTO-2038 is attached.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800.

10/04/2005 SZEWDIE1 00000054 09396470
02 FC:1402 500.00 OP

I. Real Party in Interest

The real party in interest with respect to this application is Samsung Electronics Co., Ltd., the assignee of record in this application by virtue of the assignment submitted on November 23, 1999 and recorded at reel/frame: 1010396/0163.

II. Related Appeals and Interferences

There are no prior or pending appeals, interferences or judicial proceedings known to the Appellants, the Appellants' legal representative, or the assignee which may be related to, directly effect, or be directly effected by, or have a bearing on the Board's decision in this pending appeal.

III. Status of Claims

The claims currently pending in this application are claims 1-6, 9-15 and 18-20. Claims 9, 18 and 20 have been indicated as allowed. Claim 11 has been objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form. Claims 7, 8, 16 and 17 have been withdrawn from consideration, and have subsequently been cancelled. Claims 1-6, 10, 12-15 and 19 stand finally rejected and are being appealed.

IV. Status of Amendments

An Amendment is being filed concurrently with this Brief to reduce the number of issues on appeal. The Amendment is reflected in the claims that appear in the Appendix.

V. Summary of the Claimed Subject Matter

Claims 1, 2, 3, 4 and 12-14

Claims 1, 2, 3, 4 and 12-14 are directed to a method and apparatus for coding 3-D mesh information. Figure 2 illustrates an exemplary structure of the progressive 3-D mesh information according to an embodiment of the present invention. As shown in Figure 2, the presently disclosed technique divides the 3-D mesh into a plurality of 3-D mesh layers, which are further divided into 3-D mesh objects. The 3-D mesh objects are divided into 3-D mesh components. Each of the 3-D mesh components corresponds to a different partition of the 3-D mesh. The plurality of 3-D mesh components is coded in such a manner that each is capable of being independently decoded and incrementally reproduced as a unit mesh part of the 3-D mesh. Each of the plurality of 3-D mesh components includes at least connectivity information, geometry information, and photometry information (page 6, lines 20-21). The connectivity information, geometry information and photometry information is used to reconstruct the coded 3-D mesh components (page 7, lines 14-15). A component coder which has not yet performed coding performs coding using information generated in a component coder which has already performed coding (page 7, lines 16-20). The plurality of coded 3-D mesh components are multiplexed into a compressed bit stream and transmitted (page 7, lines 12-14).

Figure 3 illustrates an exemplary apparatus for performing the above coding method. The exemplary apparatus comprises a 3-D data analyzer, a plurality of component coders and a multiplexer. The claimed features of the exemplary apparatus are recited in claims 12-14.

Claims 5, 6 and 15

These claims recite a method and apparatus for decoding a progressive 3-D mesh information by dividing a transmitted bit stream into a plurality of mesh components, which involves classifying a transmitted bit stream into one or more decoded mesh object layers and dividing each of the one or more coded mesh object layers into a plurality of mesh components (page 8, lines 26-28). Also, each of the plurality of coded mesh components is independently decoded after which the

3-D mesh is reconstructed by synthesizing the plurality of decoded 3-D mesh components. Each of the plurality of mesh components is capable of being incrementally reproduced as unit mesh parts of a 3-D mesh (page 8, line 28- page 9, line 9).

Claim 15 recites components and features (Figs. 3 and 5) of an exemplary apparatus for performing the above decoding method.

Claims 10, 11, and 19

Claims 10 and 11 are directed to a method for coding and decoding 3-D mesh information. The coding and decoding method involves extracting one or more mesh object layers from a 3-D mesh and dividing each of the mesh object layers into a plurality of independent mesh components, which is performed by element 502 of Figure 7 (page 9, lines 19-22). The coders/decoders (Fig. 7, element 503) perform the step of independently coding and transmitting the plurality of mesh components for each of the one or more mesh object layers, wherein each of the plurality of coded mesh components include information necessary such that, when decoded, is capable of being reproduced as a unit mesh part of the 3-D mesh (page 9, lines 19-24). Afterwards, a plurality of independent mesh components are obtained by decoding the plurality of independently coded and transmitted mesh components (page 9, lines 24-29).

Claim 19 recites the components (Fig. 7) that comprise an exemplary apparatus for performing the coding and decoding method.

VI. Grounds of Rejection to be Reviewed on Appeal

A. Claim 2 stands rejected under 35 U.S.C. §102(e) as allegedly being anticipated by U.S. Patent No. 6,438,266 (hereinafter the Bajaj patent).

B. Claims 1, 3-6, 10, 12-15 and 19 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over the Bajaj patent.

VII. Argument

A. The rejection of claim 2 under 35 U.S.C. §102(e) is in error insofar as the Bajaj et al. patent (US 6,438,266, hereinafter "the Bajaj patent") is not prior art.

The Bajaj patent claims the benefit of the August 27, 1998 filing date of U.S. Provisional Application No. 60/098,150 (hereinafter, "the Bajaj priority document"). A copy of the Bajaj priority document was submitted by the Appellant with the response filed December 29, 2003 (A copy is provided for the Board's convenience and is attached in the Evidence Appendix).

As can be seen on page 84 of the 150 priority document, the "format of overall data stream of compressed mesh" (shown in table 6) first includes a header, then all the encoded mesh geometry data, and then all the encoded mesh connectivity data following the encoded geometry data. With respect to the geometry data of a mesh, the '150 priority document states that "vertices are encoded in a strict order" (see, the '150 priority document, page 87, line 1), and that each layer is encoded successively (*Id.*, at about line 17). The connectivity data following the geometry data is encoded consecutively, layer-by-layer (*Id.*, pages 91-92), with triangle strips associated with their respective child contours. (*Id.*, at pages 52 and 86.)

In the Bajaj priority document, Tables 8 through 17 show the bit stream format for all of the geometry and connectivity data of the encoded 3-D mesh as performed by Bajaj. None of the tables or associated text supports the assertion in the Office Action that the Bajaj patent discloses encoding photometry information data as recited in claim 2.

It is respectfully submitted that the Bajaj priority document does not teach or suggest "dividing a transmitted bit stream into a plurality of coded mesh components, wherein the plurality of mesh components are capable of being incrementally reproduced as unit mesh parts of a 3-D mesh." As pointed out above, the Bajaj priority document does not provide support for much of the Bajaj patent disclosure relied upon in the rejection. Moreover, the separation of geometry data from

connectivity data in the bitstream of the Bajaj priority document does not teach or suggest that partitions of mesh components are capable of incremental reproduction as unit mesh parts of a 3-D mesh. Hence, in material effect the Bajaj patent 102(e) date is August 26, 1999 (not the date of the Bajaj priority document) and therefore is not prior art to the present invention by virtue of its August 26, 1999 effective filing date. The present application was filed September 15, 1999, and has a priority date of at least March 20, 1999.

Even if one were to consider, *arguendo*, that the Bajaj patent disclosed the claimed step of dividing a transmitted bit stream into a plurality of mesh components capable of being incrementally reproduced as unit mesh parts of a 3-D mesh, the Bajaj priority document makes clear that it does not support such hypothetical disclosure in the Bajaj patent, for the above reasons. In summary, the disclosure in the Bajaj patent relied upon in the rejection was added to the disclosure of the Bajaj priority document.

Furthermore, the Bajaj priority document concentrates on attempting to improve efficiency in the encoding of connectivity, geometry data and multi-resolution meshes. It is not, however, concerned with solving problems addressed by embodiments of the present invention, which include inefficiency problems related to error occurrences during transmission of mesh data, and incremental transmission and reproduction of independent mesh object layers. The conclusion of the Bajaj priority document, in fact, mentions that issues concerning treatment of lost packets during transmission should be considered for further work. (See, the Bajaj priority document, pages 69 and 82, especially page 82, lines 9-10.) Again, it is abundantly clear that the Bajaj priority document does not support the Examiner's rejection.

In contrast to the "future work" suggested by the Bajaj priority document, claim 2 recites a method capable of handling coded mesh data even when subject to damage or loss when transmitted in communication lines. As mentioned above, the Bajaj priority document does not address these problems, and more importantly, the Bajaj priority document does not disclose the features of the present invention of independent mesh object components.

Hence, it is reiterated the Section 102 prior art date for the subject matter described above would be the actual August 26, 1999 filing date of the Bajaj patent and not the August 27, 1998 filing date of the Bajaj priority application.

Appellant's claim priority to Korean Patent Application No. 1999-0009528 (hereinafter "Appellant's priority document") filed March 20, 1999 in the Korean Intellectual Property Office. A certified translation of Appellant's priority document was filed July 23, 2003 with Appellant's amendment and reply to the March 25, 2003 Office Action (A copy is provided for the Board's convenience and is attached in the Evidence Appendix). Figures 2, 3, 5 and 7 of Appellant's priority document are identical to those filed with Appellant's application. Support for the features of the claims subject to appeal can be found in the text of the Appellant's priority document corresponding to Figures 2, 3, 5 and 7.

For at least for these reasons, claim 2 is not anticipated by the Bajaj patent.

B. The rejection of claim 2 under 35 U.S.C. §102(e) is in error insofar as the Bajaj patent does not disclose the claimed subject matter.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

1. "Multiplexing" Step not disclosed.

In maintaining the rejection of claim 2, the final Office Action refers to FIG. 10 of Bajaj as disclosing step (c) of claim 2. Step (c) of claim 2 recites the step of multiplexing the plurality of coded mesh components into a compressed bit stream and transmitting the compressed bit stream. The Office Action identifies Bajaj's step

of "combining encoding" of Fig. 10 as disclosing the multiplexing step of claim 2. However, three steps before the "alternately combining encodings" step in FIG. 10, Bajaj constructs triangle strips for each triangle contour and, then, identifies the generalized triangle strips and the exceptional triangle strips. After encoding the two different types of triangle strips differently, the two types of triangle strips are then alternately combined and applied to the transmission medium. However, claim 2 recites that the plurality of coded 3-D mesh components are multiplexed, and the coded bit stream has at least three mesh components (connectivity, geometry and photometry) that are multiplexed together into a compressed bit stream. Therefore, Bajaj fails to teach multiplexing a plurality of mesh components that includes at least three types of mesh components, namely connectivity information, geometry information and photometry information as recited in the claim.

2. "Photometry Information" not disclosed.

The word "photometry" or words used to define "photometry" are not used in the Bajaj patent.

To overcome this deficiency, the Office Action states that a dictionary is used to ascertain the definition of the "photometry information." However, a copy of the definition for "photometry information" was not provided with the Office Action nor was there a citation on a Form PTO-892 to the dictionary in which the definition of "photometry information" was found. The Office Action states that based on the Examiner's understanding of "photometry information", Bajaj discloses photometry information at column 14, lines 62-65, which states:

> Object is coded as already described above. The coding may include topology, geometry, texture, normals, color, and more. Then "Instances" of the Object, which are descriptions of a mesh, its vertices and connectivity, and referenced [thereto ,[sic] are created.]

Based on the following definition of "photometry", which is the measurement of the properties of light, especially luminous intensity (American Heritage College

Dictionary, Third Edition, 2000, page 1029), the cited column and lines do not state that luminous intensity or an equivalent is coded by Bajaj. The IEEE of definition of photometry is the "1) measurement of quantities referring to radiation evaluated according to the visual effect which it produces, as based on certain conventions." (IEEE Standard Dictionary of Electrical and Electronics Terms, Second Edition, 1978, page 492). Therefore, Appellant asserts that the meaning of "photometry information" is luminous intensity and not "color and more" as disclosed by Bajaj.

Furthermore, in column 14, lines 62-65 of the Bajaj patent, it is disclosed that the object is coded, whereas claim 2 recites that it is the mesh components (MCOM) connectivity, geometry and photometry that are coded.

3. Unit Mesh Parts Decoding and Reproduction Not Disclosed

Another instance of Bajaj not disclosing all of the elements of the claimed invention is when the final Office Action refers to Figures 1, 3a-3c and 10, and column 3, lines 60+, and column 6, lines 20-30 of the Bajaj patent and asserts that these portions of the Bajaj patent teach the recited progressive 3-D mesh information coding method that includes, inter alia, the step of "(b) coding each of the plurality of mesh components wherein the plurality of code components are capable of being independently decoded and incrementally reproduced as unit mesh parts of a 3-D mesh."

When the Bajaj patent describes "progressive," it is either in the context of "progressive bit transmission" or "progressive connectivity transmission" (see the Bajaj patent, column 10, lines 62-65), both of which pertain to a multi-resolution reproduction of a 3-D mesh. For instance, in one technique described in columns 10-11 of the Bajaj patent, all the connectivity data is encoded at a first stage and not updated any further, while the geometry data is first represented in a coarse mesh level by transmitting the most significant bit, and in more detail by transmitting the bits of lower significance. The Bajaj patent also discloses a technique of progressive connectivity transmission which is used to first simplify a mesh to produce a coarse

level mesh and data pertaining to the details of the original mesh lost in the simplification process. The coarse mesh is first transmitted, restored, and then refined by the details that are transmitted later. In particular, the Bajaj patent discloses alternately performing "intra-layer decomposition" and "inter-layer decomposition." As described in columns 12-13 of the Bajaj patent, intra-layer connectivity decomposition is a mesh simplification technique in which half of the vertices in each vertex layer are decimated (i.e., every other vertex in a vertex layer is removed). In inter-layer simplification stage, the decimation involves the elimination of a whole vertex layer and re-triangulating the space between contours which were adjacent to the decimated vertex layer. The details formed with each inter-layer and intra-layer simplification are transmitted after the coarse layer mesh is transmitted. (See the Bajaj patent, columns 9-13, and the abstract).

The Bajaj patent neither discloses, nor otherwise suggests, the features of independently decoding in connection with incremental reproduction of unit mesh parts.

For at least the foregoing reasons, the §102(e) rejection of claim 2 is improper because Bajaj does not disclose all of the claimed features.

C. The rejection of claims 1, 3-6, 10, 12-15 and 19 under 35 U.S.C. §103 is in error insofar as the Bajaj patent is not prior art.

The Office Action relies on Bajaj to teach the features recited in independent claims 1, 5, 10, 12, 15 and 19. However, as presented in Section A above, the Bajaj priority document does not support the disclosure of the features relied upon by the Office Action. Accordingly, the effective filing date for the disclosure in Bajaj relied upon in the Office Action is the actual filing date of the Bajaj patent, which is August 26, 1999, which is too late to establish the Bajaj patent as a prior art reference.

Therefore, the claims 1, 3-5, 10, 12-16 and 19 are allowable for the above reasons. Claims 6 and 7, which depend upon claim 2, are also allowable for the

above reasons as well for all of the reasons stated in support withdrawing the §102(e) rejection of claim 2 above.

D. The 35 U.S.C. §103(a) rejection of claims 1, 3-6, 10, 12-16 and 19 is Improper.

1. No Suggestion or Motivation to Modify the Bajaj Patent.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, the cited references must have some suggestion or motivation either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference must teach or suggest all of the claim limitations. *In re Vaeck*, 947 F.2d 488, 20, USPQ2d 1438 (Fed. Cir. 1991).

In the rejection of claim 1, the Office Action states that Bajaj teaches that mesh components are "encoded" independently at col. 4, lines 4-5, and goes on to correctly state that Bajaj does not teach independently decoding the mesh components as recited in the claim.

In providing the motivation to support the §103(a) rejection of claim 1, the Office Action states that "It would have been obvious to one skilled in the art that "decoding" is the reverse process of encoding; therefore, by encoding the mesh component independently possibly the decoding would be independent too". (emphasis added). The use of the word "possibly" indicates that one of ordinary skill in the art may not look to decode the encoded data in the reverse process of the encoding steps. In addition, even if Bajaj can be modified to perform independent decoding there must be some suggestion or motivation in the Bajaj patent to do so. See *In re Mills*, 916 F.2d 680, 682 (Fed. Cir. 1990).

The 35 U.S.C. §103(a) rejection of claims 3-6, 10, 12-16 and 19 is also not supported by an adequate suggestion or motivation to modify Bajaj to disclose the claimed features. For instance, the Office Action states "one of skill in the art would use the decoder (18) [of Fig. 1] to decode the transmitted encoded 3-d image", but does not state how the functional steps are performed by Bajaj's decoder. Also, the Office Action provides the unsupported statement that "the decoding process in claim 5 is reversible process of the encoder" without providing a cite to the motivation relied on in Bajaj to modify Bajaj's decoding process to decode the encoded mesh components in the manner recited in the claims.

Therefore, the rejection under 35 U.S.C. §103(a) fails to provide an adequate statement of motivation of why one of ordinary skill in the art would modify the deficiencies of Bajaj to render obvious the claimed features of claims 1, 3-7, 10, 12-16 and 19.

2. Improper Procedure of making a 35 U.S.C. §103(a) rejection.

In *Graham V. Deere*, the Supreme Court laid out four factual inquiries that must be made when forming a claim rejection based on obviousness under 35 U.S.C. §103. One of the four factual inquires is ascertaining the differences between the prior art and the claims at issue.

With regard to the 35 U.S.C. §103(a) rejection of claims 3-7, 10, 12-16 and 19, the Office Action fails to ascertain the differences between the prior art and the claims at issue. The Office Action does not identify the claimed features not disclosed in Bajaj. The Office merely states how one of ordinary skill would modify the components of Bajaj to perform the functions of claims 3-7, 10, 12-16 and 19.

Therefore, the rejection of claims 3-7, 10, 12-16 and 19 under 35 U.S.C. §103(a) is improper because the Office Action does not identify the features of the claimed invention that are not taught by the applied prior art.

E. The rejection of claims 1, 3-6, 10, 12-15 and 19 under 35 U.S.C. §103 is in error insofar as the Bajaj patent does not disclose the claimed subject matter.

As stated above in Section D, to establish a *prima facie* case of obvious, three basic criteria must be met. The third and final criteria quoted from *In re Vaeck* is that "the prior art reference must teach or suggest all of the claim limitations."

1. "Coding" step not disclosed.

With respect to claims 1 and 10, the Office Action, referring again to Figures 1, 3a-3c and 10, and column 3, lines 60+ and column 6, lines 20-30 of the Bajaj patent, alleges that the Bajaj patent teaches the claimed step of "coding each of the plurality of mesh components, wherein the plurality of coded mesh components are capable of being decoded and incrementally reproduced as unit mesh parts of the 3-D mesh." Figures 1, 3a-3c, 10 do not disclose these features and the cited columns and lines of the Bajaj patent do not disclose coding mesh components capable of being decoded and incrementally reproduced. Claims 3, 4 and 11 are dependent from claims 1 and 10, respectively, and are patentable for at least the same reasons.

2. "Mesh Components" not disclosed.

Bajaj Figures 3a-3c illustrate the decomposition of an exemplary mesh in three different instances. In Fig. 3a, decomposition of a mesh using a single starting point is shown. In Fig. 3b, decomposition of a mesh where the first layer is a path is shown and, in Fig. 3c, decomposition of a mesh with branching is illustrated. When considering the Bajaj patent text that describes Figs. 3a-3c (col. 7, line 31 – col. 8, line 25), it consistently refers to layers and not to mesh components, which is a feature recited in each of the independent claims 1, 5, 10, 12, 15, and 19.

3. "Dividing" and "Demultiplexing" steps not disclosed.

With regard to the flowchart illustrated in Fig. 10 relied upon by the Examiner, none of the steps in the flow chart disclose or suggests the claimed features of 1) dividing a transmitted bit stream into a plurality of coded mesh components capable of being incrementally reproduced as unit mesh components (claim 5); 2) dividing each of the plurality of mesh object layers in to a plurality of independent mesh components (claim 10); or 3) a demultiplexer for dividing the transmitted bit stream into a plurality of coded mesh components (claim 15). Therefore, claims 5, 10 and 15 are patentable for at least the above reasons.

4. Step of "Classifying" not disclosed.

Claim 6 is also independently patentable insofar as it recites the steps of classifying transmitted bitstream into one or more decoded mesh object layers and dividing each of the one or more decoded mesh object layers into a plurality of mesh components. The Bajaj patent does not disclose or suggest classifying the transmitted bit streams into decoded mesh object layers and dividing each of the decoded mesh object layers into a plurality of mesh components.

5. Structure of "3-D Analyzer" not disclosed.

Independent claim 12 is equally patentable insofar as it recites a progressive 3-D mesh information coding apparatus that includes structure including, inter alia, a plurality of component coders for coding a plurality of mesh components. This feature of incremental reproduction as to unit mesh components is thereby distinguished in claim 12, particularly in its recitation of a plurality of coded mesh components that are capable of being decoded and incrementally reproduced as mesh unit parts of a 3-D mesh. The Office Action points to Figs. 3a and 10 of Bajaj to disclose the 3-D data analyzer, however, neither figure shows a structure that performs the functions of the 3-D data analyzer as recited in the claim. The

structural features of claim 12 are neither disclosed nor suggested by the Bajaj patent.

Claim 13, which dependent on claim 12, is equally patentable for reciting, among other things, a plurality of mesh component analyzers for again dividing each one of the one or more mesh object layers into a plurality of mesh components, thus defining a hierarchy not disclosed in the applied art, at least in the context of claim 13, when read in light of claim 12. Claim 14, which is dependent on claim 12, introduces the concept of using coding information generated in a component coder which has already performed coding. Besides not disclosing the limitations of claim 12 above, the Bajaj patent does not provide any suggestion or disclosure of the structural or functional limitations of claims 13 and 14.

6. Bajaj Priority Document Provides No Support for Rejections.

Independent claims 10 and 19 similarly recite combinations of features that are not believed taught or suggested in the Bajaj patent and the Bajaj priority document. For instance, claim 10 is directed to a method for progressively coding/decoding information of a 3-D mesh that includes, *inter alia*, a step of "extracting one or more mesh object layers from a 3-D mesh and dividing each of the mesh object layers into a plurality of independent mesh components." Claim 19 recites, *inter alia*, a 3-D mesh object layer analyzer for receiving a 3-D mesh, dividing an input 3-D mesh into one or more mesh object layers, and again dividing each mesh object layer into a plurality of independent mesh components. Therefore, the rejection of claims 10, 11 and 19 should be withdrawn for the above reasons.

7. Similar Sounding Terminology Not the Same.

In reviewing the relatively extensive prosecution history, it occurs to the undersigned that part of the difference in opinion as to the patentability of the claims lies in the fact that terms like "layer", "partition" and "progressive" are used by both the applied art and the present invention, but to mean different things.

a. Layer Structure not the same.

As illustrated in Figure 2 of the present application in an exemplary embodiment, a mesh object (MO) is divided into independent mesh object layers (MOLs), and MOLs are divided into mesh components (MCOM), which in turn are composed of connectivity, geometry and photometry for example. A basic point is that layers are independent from each other. This is in marked contrast to the Baja patent, wherein the given mesh is converted into a typical triangle strip structure by a layered decomposition such as shown in Figure 3 of the Bajaj patent. Then, vertices are defined as layers by grouping the vertices on the contour as shown in Figure 6 of the Bajaj patent. The layers are of a parent-child relationship. The layers are not separated from each other and are not independent from each other.

b. Mesh Component Structure not the same.

Similarly, in the embodiments disclosed in the present application, the 3-D mesh is divided into a plurality of mesh components and each mesh component can be encoded. That is, each 3-D mesh comprises mesh components (e.g., geometry, connectivity and photometry information) and each mesh component can be processed independently. Therefore, each mesh component is combined to form a complete 3-D mesh after independent processing. In marked contrast, in the Bajaj patent, *connectivity* progressive coding is described at columns 9-13 and the abstract. Connectivity progressive coding, as disclosed by Bajaj, means that the number of vertices is divided into two groups. One group is transmitted over an intra-layer and the other group is transmitted as an inter-layer. Unlike the present invention, the connectivity progressive coding of the Bajaj patent is for enhancing the resolution by dividing the number of vertices, transmitting/restoring the first half as intra information and adding the latter half, with respect to progressive coding. Further, as described at columns 10 and 11 of the Bajaj patent, geometry information is represented at a course level by transmitting the MSB (Most Significant Bits) and at a detail level by transmitting the lower bits.

Accordingly, the rejections of claims 1, 3-6, 10, 12-15 and 19 should be withdrawn because the teachings of Bajaj relied upon in the Office Action are not the same structures disclosed and claimed in the Appellant's application.

F. CONCLUSION

Appellants have specifically identified errors in the rejections and have explained why the rejected claims are patentable under 35 U.S.C. §102 and 35 U.S.C. §103, including specific reference to recitations in the rejected claims which are not described in the prior art relied upon in the rejection or, as appropriate, how such recitations render the claimed subject matter unobvious over the prior art.

Respectfully submitted,

Buchanan Ingersoll PC.

Date October 3, 2005 By: Martin E Miller

Martin E. Miller
Registration No. 56,022

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

Table of Contents

VIII. CLAIMS APPENDIX

The Appealed Claims

1. A progressive 3-D mesh information coding method comprising the steps of:

(a) dividing a 3-D mesh into a plurality of mesh components, wherein each of the mesh components corresponds to a different partition of the 3-D mesh;

(b) coding each of the plurality of mesh components, wherein the plurality of coded mesh components are capable of being independently decoded and incrementally reproduced as unit mesh parts of the 3-D mesh; and

(c) multiplexing the plurality of coded mesh components into a compressed bit stream and transmitting the compressed bit stream.

2. A progressive 3-D mesh information coding method comprising the steps of:

(a) dividing a 3-D mesh into a plurality of mesh components, wherein each of the mesh components corresponds to a different partition of the 3-D mesh;

(b) coding each of the plurality of mesh components; and

(c) multiplexing the plurality of coded mesh components into a compressed bit stream and transmitting the compressed bit stream, wherein each of the plurality of mesh components includes at least connectivity information, geometry information and photometry information which are necessary to reconstruct the coded mesh components themselves.

3. The progressive 3-D mesh information coding method as claimed in claim 1, wherein the step (a) comprises the substeps of:

(a1) extracting one or more mesh object layers from a 3-D mesh; and

(a2) dividing the one or more mesh object layers each into the plurality of mesh components.

4. The progressive 3-D mesh information coding method as claimed in claim 1, wherein in the step (b), each of the plurality of mesh components is coded, and information generated while a mesh component is coded is reused in the process for coding a mesh component which has not yet been coded.

5. A progressive 3-D mesh information decoding method comprising the steps of:

(a) dividing a transmitted bit stream into a plurality of coded mesh components, wherein the plurality of mesh components are capable of being incrementally reproduced as unit mesh parts of a 3-D mesh;

(b) independently decoding each of the plurality of coded mesh components; and

(c) reconstructing the 3-D mesh by synthesizing the plurality of decoded mesh components.

6. The progressive 3-D mesh information decoding method as claimed in claim 5, wherein the step (a) comprises the substeps of:

(a1) classifying the transmitted bit stream into one or more decoded mesh object layers; and

(a2) dividing each of the one or more decoded mesh object layers into a plurality of mesh components.

10. A progressive 3-D mesh information coding/decoding method comprising the steps of:

(a) extracting one or more mesh object layers from a 3-D mesh and dividing each of the mesh object layers into a plurality of independent mesh components;

(b) independently coding and transmitting the plurality of mesh components for each of the one or more mesh object layers, wherein each of the plurality of coded mesh components include information necessary such that, when decoded, is capable of being reproduced as a unit mesh part of the 3-D mesh; and

(c) obtaining a plurality of independent mesh components by decoding the plurality of independently coded and transmitted mesh components.

12. A progressive 3-D mesh information coding apparatus comprising:

a 3-D data analyzer for receiving a 3-D mesh and reconstructing the input 3-D mesh into a plurality of mesh components;

a plurality of component coders for coding the plurality of mesh components; and

a multiplexer for multiplexing the plurality of coded mesh components into a compressed bit stream, wherein the plurality of coded mesh components are capable of being decoded and incrementally reproduced as unit mesh parts of the 3-D mesh.

13. The progressive 3-D mesh information coding apparatus as claimed in claim 12, wherein the 3-D data analyzer comprises:

a 3-D mesh object layer analyzer for dividing the input 3-D mesh into one or more mesh object layers; and

a plurality of mesh component analyzers for again dividing each of one or more mesh object layers into a plurality of mesh components.

14. The progressive 3-D mesh information coding apparatus as claimed in claim 12, wherein when the plurality of component coders code the plurality of mesh components, a component coder which has not yet performed coding performs coding using coding information generated in a component coder which has already performed coding.

15. A progressive 3-D mesh information decoding apparatus for progressively decoding a bit stream decoded by and transmitted from a progressive 3-D mesh information coding apparatus, comprising:

a demultiplexer for dividing the transmitted bit stream into a plurality of coded mesh components;

a plurality of component decoders for decoding the plurality of coded mesh components, wherein the plurality of decoded mesh components are capable of being incrementally reproduced as unit mesh parts of a 3-D mesh; and

a 3-D data synthesizer for synthesizing the plurality of decoded mesh components to reconstruct the 3-D mesh.

19. A progressive 3-D mesh information coding/decoding apparatus comprising:

a 3-D mesh object layer analyzer for receiving a 3-D mesh, dividing an input 3-D mesh into one or more mesh object layers, and again dividing each mesh object layer into a plurality of independent mesh components;

a plurality of mesh component coders for independently coding and transmitting the plurality of independent mesh components, wherein each of the plurality of coded mesh components include information necessary such that, when decoded, is capable of being rendered as a unit mesh part of the 3-D mesh; and

a plurality of mesh component decoders for decoding the plurality of mesh components which are independently coded and transmitted, to obtain a plurality of independent mesh components.

IX. EVIDENCE APPENDIX

A. U.S. Provisional Application Serial No. 60/098,150, which was submitted by Appellant with the response filed December 29, 2003.

B. Certified Translation of Korean Patent Application No. 1999-0009528, which was submitted by Appellant with the amendment and reply filed on July 23, 2003.

X. RELATED PROCEEDINGS APPENDIX

Not applicable.

CERTIFICATION OF TRANSLATION

I, ***KIM, So Hee***, an employee of Y.P. LEE, MOCK & PARTNERS of The Cheonghwa Bldg., 1571-18 Seocho-dong, Seocho-gu, Seoul, Republic of Korea, hereby declare under penalty of perjury that I understand the Korean language and the English language; that I am fully capable of translating from Korean to English and vice versa; and that, to the best of my knowledge and belief, the statements in the English language in the attached translation of **the Korean Patent Application No. 1999-0009528, *Method of progressively coding/decoding 3-D mesh information and apparatus thereof*** consisting of ***26*** pages, have the same meanings as the statements in the Korean language in the original document, a copy of which I have examined.

Signed this 19th day of July, 2004

Sohee Kim

KIM, So Hee

ABSTRACT

[Abstract of the Disclosure]

A method of progressively coding/decoding 3-D mesh information, and an apparatus therefor are provided. The progressive 3-D mesh information coding method includes the steps of (a) reconstructing an input 3-D mesh into a plurality of mesh components, (b) coding each of the plurality of mesh components, and (c) multiplexing the plurality of coded mesh components into a compressed bit stream and transmitting the compressed bit stream. The method of progressively decoding the transmitted, compressed bit stream which has been coded by the coding method, includes (a) dividing the transmitted bit stream into a plurality of coded mesh components, (b) decoding each of the plurality of coded mesh components, and (c) reconstructing a 3-D mesh by synthesizing the plurality of decoded mesh components.

[Representative Drawing]

FIG. 3

SPECIFICATION

[Title of the Invention]

Method of progressively coding/decoding 3-D mesh information and apparatus thereof

[Brief Description of the Drawings]

FIG. 1 is a conceptual block diagram illustrating a conventional 3-D mesh information coding/decoding method;

FIG. 2 is a conceptual block diagram illustrating the structure of progressive 3-D mesh information according to the present invention;

FIG. 3 is a conceptual block diagram illustrating a progressive 3-D mesh information coding/decoding method according to a preferred embodiment of the present invention;

FIG. 4 is a block diagram illustrating a preferred embodiment of the 3-D data analyzer shown in FIG. 3;

FIG. 5 is a conceptual block diagram illustrating a progressive 3-D mesh information coding/decoding method according to another preferred embodiment of the present invention;

FIG. 6 is a block diagram of FIG. 5 to which a 3-D MOL synthesizer is added;

FIG. 7 is a conceptual block diagram illustrating a progressive 3-D mesh information coding/decoding method according to still another preferred embodiment of the present invention;

FIG. 8 is a block diagram of FIG. 7 to which a 3-D data synthesizer is added;

FIG. 9 illustrates an example of a simple 3-D mesh object (MO) comprised of one mesh object layer (MOL), to facilitate understanding of the present invention; and

FIG. 10 illustrates an example of three mesh components (MCOM0 through MCOM2) into which the mesh object layer (MOL) shown in FIG. 9 is divided;

In FIG. 11, (a) through (c) illustrate examples of information included in mesh components MCOM0 through MCOM2 into which a mesh object layer (MOL) is divided; and

In FIG. 12, (a) through (c) illustrate examples of a method of processing information shared by two mesh components.

[Detailed Description of the Invention]

[Object of the Invention]

[Technical Field of the Invention and Related Art prior to the Invention]

The present invention relates to coding/decoding of 3-D mesh information, and more particularly, to a progressive coding/decoding method for 3-D mesh information which is used in the field of moving picture expert group (MPEG)-4 synthetic and natural hybrid coding (SNHC) and a virtual reality modeling language (VRML), and an apparatus thereof.

Not only efficient coding of mesh data but also progressive reconstruction of transmitted mesh data is recognized as an important requisite for transmission of a 3-D object including 3-D mesh. When mesh data is damaged by a communications line error during transmission, part of the damaged mesh data can be reconstructed with already-transmitted mesh data by a progressive reconstruction technique. Thus, the amount of mesh data to be retransmitted is minimized. This progressive reconstruction technique is expected to be effectively used in future wireless communications or communications at a low transmission rate, since this technique has strong characteristics with respect to communications line errors.

FIG. 1 is a conceptual block diagram of a conventional 3-D mesh information coding/decoding apparatus. Referring to FIG. 1, a coding unit 101 is comprised of a connectivity information coder 102, a geometry information coder 103, and an entropy coder 104, and a decoding unit 112 is comprised of an entropy decoder 106, a connectivity information decoder 107, and a geometry information decoder 108.

A conventional method of compressing 3-D mesh data, which is used in MPEG, will now be described referring to FIG. 1. 3-D mesh data 100 input to the coding unit 101 includes connectivity information and geometry information, and the two types of information are coded respectively by the connectivity information coder 102 and the geometry information coder 103. Here, information 105 on a vertex structure is transmitted from the connectivity information coder 102 to the geometry information coder 103. Information compressed by the connectivity information

coder 102 and the geometry information coder 103 is converted into a compressed bit stream 111 by the entropy coder 104.

The compressed bit stream 111 is input to the decoding unit 112 and decoded as follows. The compressed bit stream 111 is divided into connectivity information and geometry information via the entropy decoder 106, and the two types of information are decoded by the connectivity information decoder 107 and the geometry information decoder 108, respectively. Similar to the coding unit 101, information 109 on a vertex structure is transmitted from the connectivity information decoder 107 to the geometry information decoder 108. A reconstructed 3-D mesh 110 can be obtained by decoded connectivity information and decoded geometry information.

As shown in FIG. 1, a 3-D mesh is transmitted in the form of a compressed bit stream on a communications line. However, since the conventional method uses the entropy coder, it has poor resistance to transmission errors which may be generated in a communications line.

Since conventional coding with respect to 3-D mesh data is accomplished in units of the entire mesh data, it is almost impossible to perform partial reconstruction before the entire bit stream is transmitted upon transmission of coded data. Also, conventional 3-D mesh coding has an inefficiency problem in that even when a very small portion of data is damaged by an error of a communications line caused upon transmission, the entire mesh data must be transmitted again. For example, an encoding method (ISO/IEC JTC1/SC29/WG11 MPEG98/W2301, MPEG-4 SNHC Verification Model 9.0) proposed by the IBM company has been used for MPEG-4 SNHC 3-D mesh coding.

[Technical Goal of the Invention]

To solve the above problems, it is an object of the present invention to provide a progressive 3-D mesh information coding/decoding method by which partial classification and partial reconstruction are possible by reconstructing a model so that it can be processed in units of parts, so that progressive picture reproduction becomes possible through progressive decoding, and transmission errors are dealt with well, and an apparatus therefor.

It is another object of the present invention to provide a progressive 3-D mesh information coding/decoding method by which independent coding and decoding are possible by dividing a model into independent step meshes or mesh components, so that progressive picture reproduction becomes possible through progressive decoding, and transmission errors are dealt with well, and an apparatus therefor.

[Structure and Operation of the Invention]

Accordingly, to achieve the first object, the present invention provides a progressive 3-D mesh information coding method which includes the steps of: (a) reconstructing a 3-D mesh into a plurality of mesh components; (b) coding each of the plurality of mesh components; and (c) multiplexing the plurality of coded mesh components into a bit stream and transmitting the bit stream.

To achieve the first object, the present invention also provides a progressive 3-D mesh information decoding method which includes the steps of: (a) dividing the transmitted bit stream into a plurality of coded mesh components; (b) decoding each of the plurality of coded mesh components; and (c) reconstructing a 3-D mesh by synthesizing the plurality of decoded mesh components.

To achieve the first object, the present invention provides a progressive 3-D mesh information coding apparatus which includes: a 3-D data analyzer for reconstructing a 3-D mesh into a plurality of mesh components; a plurality of component coders for coding the plurality of mesh components; and a multiplexer for multiplexing the plurality of coded mesh components into a bit stream.

To achieve the first object, the present invention provides a progressive 3-D mesh information decoding apparatus includes: a demultiplexer for dividing the transmitted bit stream into a plurality of coded mesh components; a plurality of component decoders for decoding the plurality of coded mesh components; and a 3-D data synthesizer for synthesizing the plurality of decoded mesh components to reconstruct a 3-D mesh.

To achieve the second object, the present invention provides another progressive 3-D mesh information coding/decoding method which includes the steps of: (a) extracting one or more independent mesh object layers from a 3-D mesh; (b) independently coding and transmitting the mesh object layers; and (c) obtaining one

or more independent mesh object layers by decoding the independently coded and transmitted mesh object layers, the method further comprising the step of (d) synthesizing the independent mesh object layers and removing redundant information to reconstruct the original 3-D mesh.

To achieve the second object, the present invention provides another progressive 3-D mesh information coding/decoding apparatus which includes: a 3-D mesh object layer analyzer for receiving a 3-D mesh and extracting one or more mesh object layers from the received 3-D mesh; one or more mesh object layer coders for independently coding and transmitting the mesh object layers; and one or more mesh object layer decoders for decoding the mesh object layers which have been independently coded and transmitted, to obtain one or more independent mesh object layers, the apparatus further including a 3-D mesh object layer synthesizer for synthesizing the independent mesh object layers and removing redundant information to reconstruct the original 3-D mesh.

To achieve the second object, the present invention provides a still another progressive 3-D mesh information coding/decoding method which includes the steps of: (a) extracting one or more mesh object layers from a 3-D mesh and dividing each mesh object layer into a plurality of independent mesh components; (b) independently coding and transmitting the plurality of mesh components; and (c) obtaining a plurality of independent mesh components by decoding the plurality of independently coded and transmitted mesh components, the method further comprising the step of (d) synthesizing the independent mesh components and removing redundant information between adjacent mesh components to reconstruct the original 3-D mesh.

To achieve the second object, the present invention provides a still another progressive 3-D mesh information coding/decoding apparatus which includes: a 3-D mesh object layer analyzer for receiving a 3-D mesh, extracting one or more mesh object layers from the received 3-D mesh, and dividing each mesh object layer into a plurality of independent mesh components; a plurality of mesh component coders for independently coding and transmitting the plurality of mesh components; and a plurality of mesh component decoders for decoding the plurality of mesh components which have been independently coded and transmitted, to obtain a

plurality of independent mesh components, the apparatus further comprising a 3-D data synthesizer for synthesizing the plurality of independent mesh components and removing redundant information between adjacent mesh components to reconstruct the original 3-D mesh.

Preferred embodiments of the present invention will now be described with reference to the attached drawings.

FIG. 2 is a conceptual block diagram illustrating the structure of progressive 3-D mesh information according to the present invention.

The present invention proposes a new mesh structure as shown in FIG. 2, to deal with a progressive 3-D mesh. Referring to FIG. 2, the 3-D mesh object (MO) can be comprised of mesh object layers (MOLs) obtained by dividing information in the mesh into layers. Here, each MOL includes one or more mesh components (MCOM). One mesh component (MCOM) includes connectivity information, geometry information, and other information such as photometry information. That is, the MO is defined as a 3-D mesh object (MO) unit to be coded, and is classified into a plurality of layers according to the various image qualities and functions of the above-described mesh information. Each of the layers is defined as MOLs. When one 3-D MOL is comprised of several independent types of mesh information (that is, connectivity components) having no connectivity therebetween using a topological surgery method, independent mesh information is synthesized or divided according to the size of data to be coded and other characteristics, and the results are defined as MCOMs.

Referring to FIG. 3, a progressive 3-D mesh information coding/decoding apparatus according to a preferred embodiment of the present invention includes a coding unit 200 and a decoding unit 209. The coding unit 200 includes a 3-D data analyzer 201, a plurality of first through N-th component coders 202, and a multiplexer (MUX) 204. The decoding unit 209 includes a demultiplexer (DMUX) 205, a plurality of first through N-th component decoders 206, and a 3-D data synthesizer 208.

Referring to FIG. 3, first, a 3-D mesh object (MO) 100 is reconstructed into a plurality of mesh components (MCOM) by the 3-D data analyzer 201, and the plurality of mesh components are input to the plurality of first through N-th

component coders 202, respectively. Here, several mesh components (MCOM) can constitute an MOL. Each of the MCOMs is compressed by a corresponding component coder 202, and the compressed bit streams are multiplexed by the MUX 204 and transmitted. Here, MOL or MCOM information 203 used in an already-operated component coder can be used in a component coder not yet operated. For example, information 203 used in the first component coder which is an upper component coder, is also transmitted to a second component coder which is a lower component coder, and can be used therein.

The compressed bit stream transmitted to the decoding unit 209 is classified into MOLs, and each MOL is again divided into mesh components (MCOMs) by the DMUX 205. The mesh components are decoded by the plurality of first through N-th component decoders 206. Also in the decoding unit 209, information 207 generated by an already-operated component decoder is reused in a component decoder not yet operated. For example, information 207 generated by the first component decoder which is an upper component decoder, is transmitted to a second component decoder which is a lower component decoder. The decoded mesh components (MCOMs) are reconstructed by the 3-D data synthesizer 208 into a 3-D mesh 110.

When the 3-D MO is first classified into one or more MOLs and then divided into MCOMs, the 3-D data analyzer 201 can be constituted as shown in FIG. 4.

Referring to FIG. 4, a preferred embodiment of the 3-D data analyzer 201 of FIG. 3 includes a 3-D MOL analyzer 301 and a plurality of first through n-th MCOM analyzers 303.

When the 3-D MO 100 is input to the 3-D data analyzer 300, the 3-D MOL analyzer 301 extracts MOL1 through MOLn 302 from the input 3-D MO. Then, the extracted MOL1 through MOLn 302 are each divided into mesh components 304 by the first through n-th MCOM analyzers 303. The mesh components 304 are output from the data analyzers. Each mesh component 304 is input to a corresponding component coder among (1-1)th through (1-m)th component coders, (2-1)th through (2-m) component coders, etc.

Each of the MCOM analyzers uses information generated by its upper MCOM analyzer, and the component coders for one MCOM analyzer uses information

generated by an above component coder corresponding to the same MCOM analyzer. However, information 305 used in an arbitrary MCOM analyzer may not be used in another MCOM analyzer. In this case, an independent mesh object layer information coding/decoding method can be performed in a structure shown in FIG. 5.

Referring to FIG. 5, a progressive 3-D mesh information coding/decoding apparatus according to another embodiment of the present invention includes a 3-D MOL analyzer 401, and a plurality of first through n-th independent coders/decoders 403.

In this embodiment shown in FIG. 5, information generated by any one coder is not used in the other coders. That is, after MOL1 through MOLn 402 are generated from the 3-D mesh object (MO) 100 by the 3-D MOL analyzer 401, they are compressed, transmitted, decoded and reconstructed respectively by the independent coders/decoders 403. Here, each of the coders and decoders includes an MCOM analyzer, so that each can divide a MOL into MCOMs and code/decode them. Information 404 decoded by the decoders is independent MOL data, so a 3-D mesh 405 is reconstructed by simply collecting this information. However, in this case, the reconstructed 3-D mesh 405 also includes extra information which was generated when the 3-D MOL analyzer 401 classified the 3-D mesh object (MO) 100 into the MOL1 through MOLn 402.

Referring to FIG. 6, the preferred embodiment shown in FIG. 5 further includes a 3-D MOL synthesizer 406. The 3-D MOL synthesizer 406 is added to obtain a reconstructed 3-D mesh 110 which is the same as the original 3-D mesh MO 100 by removing the additional information included in the reconstructed 3-D mesh 405. The 3-D MOL synthesizer 406 collects information 404 decoded by the decoders and removes redundant information, thereby reconstructing the original 3-D mesh 110.

Meanwhile, the component coders in FIG. 4 share information, but information 306 used in one component coder may not be used in the other component coders. In this case, an independent component mesh information coding/decoding method can be performed in a structure shown in FIG. 7.

Referring to FIG. 7, a progressive 3-D mesh information coding/decoding apparatus according to still another embodiment of the present invention includes a 3-D data analyzer 501, and a plurality of first through n-th independent MCOM coders/decoders 503.

In the embodiment shown in FIG. 7, information used in an already-operated coder will not be used in a coder not yet operated. That is, the 3-D mesh MO 100 is divided into mesh components MCOM1 through MCOMn 502 by the 3-D data analyzer 501 which includes a 3-D MOL analyzer and a plurality of MCOM analyzers, and the mesh components 502 are then compressed, transmitted, decoded and reconstructed by the independent first through n-th MCOM coders/decoders 503, respectively. Information 504 decoded by the decoders is independent mesh component data, so reconstruction of a 3-D mesh 505 is accomplished by simply collecting this data. However, when the 3-D data analyzer 501 divides one mesh object layer into a plurality of mesh components, the mesh components share boundary information such as an edge, the attributes of the coordinates of points, or those of the units of points. Thus, much redundant information exists in the reconstructed 3-D mesh 505.

Referring to FIG. 8, still another preferred embodiment shown in FIG. 7 further includes a 3-D data synthesizer 506. The 3-D data synthesizer 506 is added as shown in FIG. 8 to increase coding efficiency by removing redundant information included in the reconstructed 3-D mesh 505. The 3-D data synthesizer 506 collects information 504 decoded by the decoders and removes redundant information, thereby reconstructing the original 3-D mesh 110.

FIG. 9 illustrates an example of a simple 3-D MO comprised of one MOL to facilitate understanding of the present invention. FIG. 10 illustrates an example in which the 3-D MOL shown in FIG. 9 is divided into three mesh components (MCOM0 through MCOM2). Information (e.g., an edge, the coordinates of points, the attributes of point units, etc.) corresponding to the boundary portion is shared by adjacent mesh components. Accordingly, the next mesh component can be decoded/coded by reusing some or all of the information generated while a mesh component is decoded/coded.

In FIG. 11, (a) through (c) illustrate examples of information included in each mesh component when an MOL is divided into mesh components MCOM0 through MCOM2.

In FIG. 11, (a) shows the sharing of information in the boundary portion between mesh components when an MOL is divided into mesh components MCOM0 through MCOM2. Some or all of information existing in the boundary portion is copied, so that adjacent mesh components can share the same information. In FIG. 11, (b) shows mesh components MCOM0 through MCOM2 each having information shared by adjacent mesh components. When the redundancy of shared information in each mesh component is allowed as shown in (b) of FIG. 11, all of the mesh components can be independently processed, but compression efficiency is degraded.

Meanwhile, in FIG. 11, (c) shows mesh components MCOM0 through MCOM2 when only a mesh component to be first coded has information shared by adjacent mesh components. When there is information shared by an already-coded mesh component, information generated by the already-coded mesh component is used for a mesh component not yet coded, so that no redundant information is generated. However, the respective mesh components are not independently processed.

In FIG. 12, (a) through (c) illustrate examples of a method of processing information shared by two mesh components.

In FIG. 12, (a) shows two mesh components which are separated independently. Here, the two mesh components share information such as the same sides, the same points and the attributes of the points. Each type of information is copied to two mesh components, and the two mesh components are separated and then independently processed. While the two mesh components are processed, information copied to each mesh component is considered to be different. Hence, when information independently decoded by a decoding portion is collected and used to reconstruct a mesh object layer, the original MOL is transformed, and consequently, the size of a file increases. Referring to FIG. 5 or 7, independent mesh components as shown in (a) of FIG. 12 are independently processed by independent coders/decoders.

In FIG. 12, (b) shows two mesh components which are not completely separated from each other but only part of the shared information is redundantly coded. A preceding mesh component has all of the shared information, but the next mesh component has only part of the shared information, that is, information on the same points as those of the preceding mesh component. In this case, the same shared point is defined by each of two or more mesh components, but the same point in each is recognized as an identical point. Thus, the original MOL is kept after the mesh components are restored. Also, when the preceding mesh component has not been restored, the boundary portion between the preceding mesh component and the next mesh component can be restored due to restoration of the next mesh component. Referring to FIG. 3, when information generated by an already-operated component coder is used in a component coder not yet operated, the information is part of the shared information. The component coder not yet operated redundantly codes information on the same points shared by the already-operated component coder.

In FIG. 12, (c) shows an example where only a preceding mesh component has shared information, and the next mesh component uses all of information generated by the preceding mesh component. In this case, no information is redundantly coded, so that compression efficiency is increased. However, since each mesh component is not independent, it is impossible to restore the boundary portion due to restoration of the next mesh component when the preceding mesh component is not restored. Referring to FIG. 3, when information generated by an already-operated component coder is used in a component coder not yet operated, the information corresponds to the whole of shared information. The component coder not yet operated does not redundantly code all information shared by the already-operated component coder.

A grammar table of ISO/IEC JTC1/SC29/WG11 MPEG-4 SNHC 3-D mesh coding, for accomplishing coding of progressive 3-D mesh information according to the present invention is as follows.

▸ **3D_Mesh_Object**

MO_start_code: This is a unique 16-bit code which is used for synchronization. The value of this code is always '0000 0000 0010 0000'.

▸ **3D_Mesh_Object_Layer**

MOL_start_code: This is a unique 16-bit code which is used for synchronization. The value of this code is always '0000 0000 0011 0000'.

mol_id: This unsigned 8-bit integer indicates a unique identifier for the mesh object layer (MOL). A value of 0 indicates a base layer, and a value larger than 0 indicates a refinement layer. The first 3D_Mesh_Object_Layer immediately behind a 3D_Mesh_Object_Header must have mol_id=0, and subsequent 3D_Mesh_Object_Layers within the same 3D_Mesh_Object must have mol_id>0.

N_Vertices denotes the number of vertices in the current resolution of the 3-D mesh, which is used to reduce the number of calculations.

N_Triangles denotes the number of triangles in the current resolution of the 3-D mesh, which is used to reduce the number of calculations.

N_Edges denotes the number of edges in the current resolution of the 3-D mesh, which is used to reduce the number of calculations.

▸ **3D_Mesh_Object_Base_Layer**

MOBL_start_code: This is a unique 16-bit code which is used for synchronization. The value of this code is always '0000 0000 0011 0001'.

mol_id: This 8-bit unsigned integer which indicates a unique identifier for the mesh object component (MCOM).

last_component: This boolean value indicates if there are more connected components to be decoded. If last_component is true, then the last component has been decoded. Otherwise, there are more components to be decoded. This field is arithmetically coded.

▸ **3D_Mesh_Object_Header**

ccw: This boolean value indicates if the vertex ordering of the decoded faces follows a counterclockwise order.

convex: This boolean value indicates if the model is convex.

solid: This boolean value indicates if the model is solid.

creaseAngle: This 6-bit unsigned integer indicates the crease angle.

▸ **coord_header**

coord_binding: This 2-bit unsigned integer indicates binding of vertex coordinates to the 3-D mesh. The only admissible value is '01'.

coord-bbox: This boolean value indicates whether a bounding box is provided for the geometry. If no bounding box is provided, a default is used.

coord_xmin, coord_ymin, and coord-zmin: These floating-point values indicate the lower left corner of the bounding box having geometry.

coord_size: This floating-point value indicates the size of the bounding box.

coord_quant: This 5-bit unsigned integer denotes the quantization step used for geometry.

coord_pred_type: This 2-bit unsigned integer denotes the type of prediction used to reproduce the vertex coordinates of the mesh.

[Table 1]

coord_pred_type	prediction type
00	no prediction
01	invalid
10	parallelogram prediction
11	polygon prediction

coord_nlambda: This 2-bit unsigned integer denotes the number of ancestors used to predict geometry. Admissible values of coord_nlambda are 1 and 3. Table 2 shows admissible values as a function of coord_pred_type.

[Table 2]

coord_pre_type	coord_nlambda
00	1
01	invalid
10	3
11	1

coord_lambda: This unsigned integer indicates the weight given to an ancestor for prediction. The number of bits used for this field is equal to coord_quant+3.

▸ **normal_header**

normal_binding: This 2-bit unsigned integer indicates the binding of normals to the 3D mesh. The admissible values are described in Table 3.

[Table 3]

normal_binding	binding
00	no normals are coded
01	one normal is coded per vertex
10	one normal is coded per face
11	one normal is coded per corner

normal_bbox: This boolean value should always be false ('0').

normal_quant: This 5-bit unsigned integer indicates the quantization step used for normals.

normal_pred_type: This 2-bit unsigned integer indicates how normal values are predicted.

[Table 4]

normal_pred_type	prediction type
00	no prediction
01	tree prediction
10	parallelogram prediction
11	invalid

[Table 5]

normal_binding	normal_pred_type
00	ignored
01	10
10	01
11	01

normal_nlambda: This 2-bit unsigned integer indicates the number of ancestors used to predict normals. Admissible values of normal_nlambda are 1 and 3. Table 6 shows admissible values as a function of normal_pred_type.

[Table 6]

normal_pred_type	normal_nlambda
00	1
01	1
10	3
11	1

normal_lambda: This unsigned integer indicates the weight given to an ancestor for prediction. The number of bits used for this field is equal to normal_quant+3.

▸ **color_header**

color_binding: This 2-bit unsigned integer indicates the binding of colors to the 3D mesh. The admissible values are described in Table 7.

[Table 7]

color_binding	binding
00	no colors are coded
01	one color is coded per vertex

10	one color is coded per face
11	one color is coded per corner

color_bbox: This boolean value indicates if a bounding box for colors is given.

color_rmin, color_gmin and color_bmin: These floating point values give the position of the lower left corner of the bounding box in RGB space.

color_size: This floating point value gives the size of the color bounding box.

color_quant: This 5-bit unsigned integer indicates the quantization step used for colors.

color_pred_type: This 2-bit unsigned integer indicates how colors are predicted.

[Table 8]

color_pred_type	prediction type
00	no prediction
01	tree prediction
10	parallelogram prediction
11	invalid

[Table 9]

color_binding	color_pred_type
00	ignored
01	10
10	01
11	01

color_nlambda: This 2-bit unsigned integer indicates the number of ancestors used to predict colors. Admissible values of color_nlambda are 1 and 3. Table 10 shows admissible values as a function of color_pred_type.

[Table 10]

color_pred_type	color_nlambda
00	1
01	1
10	3
11	1

color_lambda: This unsigned integer indicates the weight given to an ancestor for prediction. The number of bits used for this field is equal to color_quant+3.

▸ **texCoord_header**

texCoord_binding: This 2-bit unsigned integer indicates the binding of textures to the 3D mesh. The admissible values are described in Table 11.

[Table 11]

texCoord_binding	binding
00	no textures are coded
01	one texture is coded per vertex
10	one texture is coded per face
11	one texture is coded per corner

texCoord_bbox: This boolean value indicates if a bounding box for textures is given.

texCoord_umin and texCoord_vmin: These two floating point values give the position of the lower left corner of the bounding box in 2D space.

texCoord_size: This floating point value gives the size of the texture bounding box.

texCoord_quant: This 5-bit unsigned integer indicates the quantization step used for textures.

texCoord_pred_type: This 2-bit unsigned integer is always '10' if texCoord_binding is '01', and '01' otherwise.

texCoord_nlambda: This 2-bit unsigned integer indicates the number of ancestors used to predict textures. Admissible values of texCoord_nlambda are 1 and 3. Table 12 shows admissible values as a function of texCoord_pred_type.

[Table 12]

texCoord_pred_type	texCoord_nlambda
00	1
01	1
10	3
11	1

texCoord_lambda: This unsigned integer indicates the weight given to an ancestor for prediction. The number of bits used for this field is equal to texCoord_quant+3.

▸ **Cgd_header**

N-Proj_Surface_Spheres is the number of Projected Surface Spheres. Typically, this number is equal to 1.

x_coord_Center_Point is the x-coordinate of the center point (typically the gravity point of the object) of the Projected Surface Sphere.

y_coord_Center_Point is the y-coordinate of the center point (typically the gravity point of the object) of the Projected Surface Sphere.

z_coord_Center_Point is the z-coordinate of the center point (typically the gravity point of the object) of the Projected Surface Sphere.

Normalized_Screen_Distance_Factor indicates where the virtual screen is placed, in relation to the radius of the projected surface sphere. The distance between the center point of the projected surface sphere and the virtual screen is equal to Radius/(Normalized_Screen_Distance_Factor+1). Note that Radius is specified for each Projected Surface Sphere, while Normalized_Screen_Distance_Factor is specified only once.

Radius is the radius of the Projected Surface Sphere.

Min_Proj_Surface is the minimal projected surface value on the corresponding Projected Surface Sphere. This value is often (but not necessarily) equal to one of the Proj_Surface values.

N_Proj_Points is the number of points on the Projected Surface Sphere in which the projected surface will be transmitted. For all other points, the projected surface is determined by linear interpolation. N_Proj_Points is typically small (e.g., 20) for the first Projected Surface Sphere and very small (e.g., 3) for additional Projected Surface Spheres.

Sphere_Point_Coord indicates the index of the point position in an octahedron.

Proj_Surface is the projected surface in the point specified by Sphere_Point_Coord.

▸ **vertex_graph**

vg_simple: This boolean value indicates if the current vertex graph is simple. A simple vertex graph does not contain any loops. This field is arithmetically coded.

vg_last: This boolean value indicates if the current run is the last run starting from the current branching vertex. This field is not coded for the first run of each branching vertex, i.e., when the skip_last variable is true. When not coded the value of vg_last for the current vertex run is considered to be false. This field is arithmetically coded.

vg_forward_run: This boolean flag indicates if the current run is a new run. If it is not a new run, it must be a run previously traversed, indicating a loop in the graph. This field is arithmetically coded.

vg_loop_index: This unsigned integer indicates the index of the current run to which the current loop is connected. Its unary representation (see Table 13) is arithmetically coded. If the variable openloops is equal to vg_loop_index, the trailing '1' in the unary representation is omitted.

[Table 13]

vg_loop_index	unary representation
0	1
1	01
2	001
3	0001
4	00001
5	000001
6	0000001
...	
openloops-1	openloops-1 0's

vg_run_length: This unsigned integer indicates the length of the current vertex run. Its unary representation (see FIG. 14) is arithmetically coded.

[Table 14]

vg_run_length	unary representation
1	1
2	01
3	001
4	0001

5	00001
6	000001
7	0000001
8	00000001
n	n-1 0's followed by 1

vg_leaf: This boolean flag indicates if the last vertex of the current run is a leaf vertex. If it is not a leaf vertex, it is a branching vertex. This field is arithmetically coded.

vg_loop: This boolean flag indicates if the leaf of the current run connects to a branching vertex of the graph, indicating a loop. This field is arithmetically coded.

▸ **triangle_tree**

tt_run_length: This unsigned integer indicates the length of the current triangle run. Its unary representation (see Table 15) is arithmetically coded.

[Table 15]

tt_run_length	unary representation
1	1
2	01
3	001
4	0001
5	00001
6	000001
7	0000001
8	00000001
n	n-1 0's followed by 1

tt_leaf: This boolean flag indicates if the last triangle of the current run is a leaf triangle. If it is not a leaf triangle, it is a branching triangle. This field is arithmetically coded.

triangulated: This boolean value indicates if the current component contains triangles only. This field is arithmetically coded.

marching_triangle: This boolean value is determined by the position of the triangle in the triangle tree. The value marching_triangle=0 if the triangle is a leaf or branching triangle, and marching_triangle=1 otherwise.

marching_pattern: This boolean flag indicates the marching pattern of edges inside a triangle run. A "0" stands for a march to the left, and a 1 for a march to the right. This field is arithmetically coded.

polygon_edge: This boolean flag indicates whether the base of the current triangle is an edge that should be kept when reconstructing the 3D mesh object. If the base of the current triangle is not to be kept, it is discarded. This field is arithmetically coded.

▸ **triangle**

coord_bit: This boolean value indicates the value of a geometry bit. This field is arithmetically coded.

coord_heading_bit: This boolean value indicates the value of a heading geometry bit. This field is arithmetically coded.

coord_sign_bit: This boolean value indicates the sign of a geometry sample. This field is arithmetically coded.

coord_trailing_bit: This boolean value indicates the value of a trailing geometry bit. This field is arithmetically coded.

normal_bit: This boolean value indicates the value of a normal bit. This field is arithmetically coded.

normal_heading_bit: This boolean value indicates the value of a heading normal bit. This field is arithmetically coded.

normal_sign_bit: This boolean value indicates the sign of a normal sample. This field is arithmetically coded.

normal_trailing_bit: This boolean value indicates the value of a normal trailing bit. This field is arithmetically coded.

color_bit: This boolean value indicates the value of a color bit. This field is arithmetically coded.

color_heading_bit: This boolean value indicates the value of a heading color bit. This field is arithmetically coded.

color_sign_bit: This boolean value indicates the sign of a color sample. This field is arithmetically coded.

color_trailing_bit: This boolean value indicates the value of a trailing color bit. This field is arithmetically coded.

texCoord_bit: This boolean value indicates the value of a texture bit. This field is arithmetically coded.

texCoord_heading_bit: This boolean value indicates the value of a heading texture bit. This field is arithmetically coded.

texCoord_sign_bit: This boolean value indicates the sign of a texture sample. This field is arithmetically coded.

texCoord_trailing_bit: This boolean value indicates the value of a trailing texture bit. This field is arithmetically coded.

▸ **3D_Mesh_Object_Forest_Split**

MOFS_start_code: This is a unique 32-bit code that is used for synchronization. The value of this code is always '0000 0000 0011 0010'.

mofs_id: This 8-bit unsigned integer specifies a unique identifier for the forest split component.

pre_smoothing: This boolean value indicates whether the current forest split operation uses a pre-smoothing step to globally predict vertex positions.

pre_smoothing_n: This integer value indicates the number of iterations of the pre-smoothing filter.

pre_smoothing_lambda: This floating point value is the first parameter of the pre-smoothing filter.

pre_smoothing_mu: This floating point value is the second parameter of the pre-smoothing filter.

post_smoothing: This boolean value indicates whether the current forest split operation uses a post-smoothing step to remove quantization artifacts.

post_smoothing_n: This integer value indicates the number of iterations of the post-smoothing filter.

post_smoothing_lambda: This floating point value is the first parameter of the post-smoothing filter.

post_smoothing_mu: This floating point value is the second parameter of the post-smoothing filter.

sharp_edges: This boolean value indicates if data that marks smoothing discontinuity edges, has been included in the bitstream. If sharp_edges==0 no edge is marked as a smoothing discontinuity edge. If smoothing discontinuity edges are marked, then both the pre-smoothing and post-smoothing filters take them into account.

fixed_vertices: This boolean value indicates if data not moving during the smoothing process has been included in the bitstream. If fixed_vertices==0, none of the vertices are allowed to move. If fixed vertices are marked, then both the pre-smoothing and post-smoothing filters take them into account.

edge_mark: This boolean value indicates if a corresponding edge is marked as a smoothing discontinuity edge.

vertex_mark: This boolean value indicates whether a corresponding vertex is fixed.

tree_edge: This boolean value indicates if an edge should be added to the forest built so far.

other_update: This boolean value indicates whether updates for vertex coordinates and properties associated with faces not incident to any tree of the forest, follow in the bitstream.

3D_Mesh_Object

3D_Mesh_Object () {		
3D_MO_start_code	16	uimsbf

3D_Mesh_Object_Header ()		
do {		
3D_Mesh_Object_Layer ()		
} while (nextbits_bytealigned ()==3D_MOL_start_code)		
}		

3D_Mesh_Object_Header

3D_Mesh_Object_Header () {		
Ccw	1	blsbf
Convex	1	blsbf
Solid	1	blsbf
CreaseAngle	6	uimsbf
Coord_header ()		
Normal_header ()		
Color_header ()		
TexCoord_header ()		
cgd_data	1	blsbf
if (cgd_data==1)		
cgd_header ()		
}		

3D_Mesh_Object_Layer

3D_Mesh_Object_Layer () {		
3D_MOL_start_code	16	uimsbf
Mol_id	8	uimsbf
if (cgd_data==1) {		
N_Vertices	24	uimsbf
N_Triangles	24	uimsbf
N_Edges	24	uimsbf
}		
if (mol_id=='00000000')		
3D_Mesh_Object_Base_Layer ()		
else		
3D_Mesh_Object_Forest_Split ()		
}		

3D_Mesh_Object_Base_Layer

3D_Mesh_Object_Base_Layer ()		
do {		
3D_MOBL_start_code	16	uimsbf
mobl_id	8	uimsbf
start_qcoder ()		
do {		
connected_component ()		

last_component		bac
} while (!last_component)		
} while (nextbits_bytealigned ()==3D_MOBL_start_code)		
}		

coord_header

coord_header () {		
coord_binding	2	uimsbf
coord_bbox	1	blsbf
if (coord_bbox) {		
coord_xmin	32	ieeefp
coord_ymin	32	ieeefp
coord_zmin	32	ieeefp
coord_size	32	ieeefp
}		
coord_quant	5	uimsbf
coord_pred_type	2	uimsbf
if (coord_pred_type=='10') {		
coord_nlambda	2	uimsbf
for (i=1;i<coord_nlambda;i++)		
coord_lambda		

	coord_quant+3	
}		
}		

normal_header

normal_header () {		
normal_binding	2	uimsbf
if (normal_binding !='00'){		
normal_bbox	1	blsbf
normal_quant	5	uimsbf
normal_pred_type	2	uimsbf
if (normal_pred_type=='10') {		
normal_nlambda	2	uimsbf
for (i=1;i<normal_nlambda;i++)		
normal_lambda	normal_quant+3	
}		
}		
}		

color_header

color_header () {		
color_binding	2	uimsbf
if (color_binding!='00') {		

color_bbox	1	blsbf
if (color_bbox) {		
color_rmin	32	ieeefp
color_gmin	32	ieeefp
color_bmin	32	ieeefp
color_size	32	ieeefp
}		
color_quant	5	uimsbf
color_pred_type	2	uimsbf
if (color_pred_type=='10') {		
color_nlambda	2	uimsbf
for (i=1;i<color_nlambda;i++)		
color_lambda	color_quant+3	
}		
}		
}		

texCoord_header

texCoord_header () {		
texCoord_binding	2	uimsbf
if (texCoord_binding!='00') {		

texCoord_bbox	1	blsbf
if (texCoord_bbox) {		
texCoord_umin	32	ieeefp
texCoord_vmin	32	ieeefp
texCoord_size	32	ieeefp
}		
texCoord_quant	5	uimsbf
texCoord_pred_type	2	uimsbf
if (texCoord_pred_type=='10'){		
texCoord_nlambda	2	uimsbf
for (I=1;i<texCoord_nlambda;i++)		
texCoord_lambda	texCoord_quant+3	
}		
}		
}		

cgd_header

cgd_header () {		
N_Proj_Surface_Spheres	4	uimsbf
if (N_Proj_Surface_Spheres<>0) {		
x_coord_Center_Point	32	ieeefl

y_coord_Center_Point	32	ieeefl
z_coord_Center_Point	32	ieeefl
Normalized_Screen_Distance_Factor	8	uimsbf
for (I=1;i<=N_Proj_Surface_Spheres;i++) {		
Radius	32	ieeefl
Min_Proj_Surface	32	ieeefl
N_Proj_Points	8	uimsbf
for (j=1;j<=N_Proj_Points;j++) {		
Sphere_Point_Coord	11	uimsbf
Proj_Surface	32	ieeefl
}		
}		
}		
}		

connected_component

connected_component () {		
vertex_graph ()		
triangle_tree ()		
triangle_data ()		
}		

vertex_graph

vertex_graph () {		
vg_simple	0-16	bac
depth=0		
skip_last=0		
openloops=0		
do {		
do {		
if (!skip_last) {		
vg_last	0-16	bac
if (openloops>0) {		
vg_forward_run	0-	bac
if (!vg_forward_run) {		
openloop–		
if (openloops>0)		
vg_loop_index	0-	bac
break		
}		
}		

}		
vg_run_length	0-	uac
vg_leaf	0-16	bac
if (vg_leaf&&!vg_simple) {		
vg_loop	0-16	bac
if (vg_loop)		
openloops++		
}		
} while (0)		
if (vg_leaf==(vg_last&!skip_last))		
if (vg_last&!skip_last)		
depth--		
else		
depth++		
skip_last=!vg_leaf		
} while (depth >= 0)		
}		

triangle_tree

triangle_tree () {		
depth = 0		
ntriangles = 0		
do {		
tt_run_length	0-16	bac
ntriangles += tt_run_length		
tt_leaf	0-16	bac
if (tt_leaf)		
depth--		
else		
depth++		
} while (depth >= 0)		
}		

triangle_data

triangle_data () {		
triangulated	0-16	bac
root_triangle ()		
for (i=1; i<ntriangles;i++)		
triangle ()		
}		

root_triangle

root_triangle () {		
if (marching_triangle)		
marching_pattern	0-16	bac
root_coord ()		
root_normal ()		
root_color ()		
root_texCoord ()		
}		

root_coord () {		
root_coord_sample ()		
coord_sample ()		
coord_sample ()		
}		

root_normal () {		
if (normal_binding != '00') {		
root_normal_sample ()		
if (normal_binding != '10') {		
normal_sample ()		
normal_sample ()		

```
                    }
                }
}
```

```
root_color () {
        if (color_binding != '00') {
                root_color_sample ()
                if (color_binding != '10') {
                        color_sample ()
                        color_sample ()
                }
        }
}
```

```
root_texCoord () {
        if (texCoord_binding != '00') {
                root_texCoord_sample ()
                texCoord_sample ()
                texCoord_sample ()
        }
```

}		

triangle

triangle () {		
if (marching_triangle)		
marching_pattern	0-16	bac
if (!triangulated)		
polygon_edge	0-16	bac
coord ()		
normal ()		
color ()		
texCoord ()		
}		

coord () {		
if (!visited)		
coord_sample ()		
}		

normal () {		
if (normal_binding == '01') {		
if (!visited)		

```
                    normal_sample ()
            }
            else if (normal_binding == '10') {
                    if (triangulated || polygon_edge)
                            normal_sample ()
            }
            else if (normal_binding == '11') {
                    if (triangulated || polygon_edge) {
                            normal_sample ()
                            normal_sample ()
                    }
                    normal_sample ()
            }
}
```

```
color () {
            if (color_binding == '01') {
                    if (!visited)
                            color_sample ()
            }
            else if (color_binding == '10') {
                    if (triangulated || polygon_edge)
```

```
                                        color_sample ()
            }
            else if (color_binding == '11') {
                        if (triangulated || polygon_edge) {
                                        color_sample ()
                                        color_sample ()
                        }
                        color_sample ()
            }
}
```

```
texCoord () {
            if (texCoord_binding == '01') {
                        if (!visited)
                                    texCoord_sample ()
            }
            else if (texCoord_binding == '11') {
                        if (triangulated || polygon_edge) {
texCoord_sample ()
texCoord_sample ()
```

}		
texCoord_sample ()		
}		
}		

coord_root_sample () {		
for (i=0; i<3; i++)		
for (j=0; j<coord_quant; j++)		
coord_bit	0-1	bac
}		

normal_root_sample () {		
for (i=0; i<1; i++)		
for (j=0; j<normal_quant; j++)		
normal_bit	0-1	bac
}		

color_root_sample () {		
for (i=0; i<3; i++)		
for (j=0; j<color_quant; j++)		
color_bit	0-1	bac
}		

texCoord_root_sample () {		
for (i=0; i<2; i++)		
for (j=0; j<texCoord_quant; j++)		
texCoord_bit	0-1	bac
}		

coord_sample () {		
for (i=0; i<3; i++) {		
j=0		
do {		
coord_leading_bit	0-16	bac
j++		
} while (j<coord_quant && !coord_leading_bit)		
if (coord_leading_bit) {		
coord_sign_bit	0-1	bac
do {		
coord_trailing_bit		
} while (j<coord_quant)		
}		

}		
}		

normal_sample () {		
for (i=0; i<1; i++) {		
j=0		
do {		
normal_leading_bit	0-16	bac
j++		
} while (j<normal_quant && !normal_leading_bit)		
if (normal_leading_bit) {		
normal_sign_bit	0-1	bac
do {		
normal_trailing_bit		
} while (j<normal_quant)		
}		
}		
}		

color_sample () {		
for (i=0; i<3; i++) {		
j=0		
do {		
color_leading_bit	0-16	bac
j++		
} while (j<color_quant && !color_leading_bit)		
if (color_leading_bit) {		
color_sign_bit	0-1	bac
do {		
color_trailing_bit		
} while (j<color_quant)		
}		
}		
}		

texCoord_sample () {		
for (i=0; i<2; i++) {		
j=0		
do {		

texCoord_leading_bit	0-16	bac
j++		
} while (j<texCoord_quant && !texCoord_leading_bit)		
if (texCoord_leading_bit) {		
texCoord_sign_bit	0-1	bac
do {		
texCoord_trailing_bit		
} while (j<texCoord_quant)		
}		
}		
}		

3D_Mesh_Object_Forest_Split

3D_Mesh_Object_Forest_Split () {		
do {		
3D_MOFS_start_code	16	uimsbf
mofs_id	8	uimsbf
pre_smoothing	1	blsbf
if(pre_smoothing)		

pre_smoothing_parameters ()	1	blsbf
post_smoothing		
if(post_smoothing)		
post_smoothing parameters()		
start_qcoder()		
sharp_edges	1	blsbf
if(sharp_edges)		
edge_marks()		
fixed_vertices	1	blsbf
if(fixed_vertices)		
vertex_marks()		
for each connected component {		
fs_pre_update()		
fs_post_update()		
}		
} while (nextbits_bytealigned()==3D_MOFS_start_code)		
}		

pre_smoothing_parameters() {		
pre_smoothing_n	8	nimsbf

pre_smoothing_lambda	32	ieefl
pre_smoothing_mu	32	ieefl
}		

post_smoothing_parameters() {		
post_smoothing_n	8	nimsbf
post_smoothing_lambda	32	ieefl
post_smoothing_mu	32	ieefl
}		

edge_marks () {		
for each edge		
edge_mark	}	bac
}		

vertex_marks () {		
for each vertex		
vertex_mark	}	bac
}		

fs_pre_updata() {		
forest()		
for each tree in forest {		

triangle_tree()		
for each vertex in tree loop		
visted = 1		
triangle_data()		
}		
}		

forest () {		
for each edge		
if (creates no loop in forest)		
tree_edge	1	bac
}		

fs_post_updata() {		
for each tree in forest {		
for each vertex in tree loop		
visted = 0		
tree_loop_property_update()		
}		
other_update	1	blsf
if(other_updata)		
other_property_updata()		

```
}
```

```
tree_loop_property_update () {
        for each triangle incident to tree {
            coord_update()
            normal_update()
            color_update()
            texCoord_update()
        }
}
```

```
other_property_update () {
    for each triangle no incident to any tree in forest {
        coord_update()
        normal_update()
        color_update()
        texCoord_update()
    }
}
```

```
coord_update() {
        if (!visited)
```

coord_sample()		

normal_updata() {		
if (normal_binding=='01') {		
if (!visited)		
normal_sample()		
}		
else if (normal_binding =='10') {		
normal_sample()		
}		
else if (normal_binding == '11') {		
if (1st corner adjacent to tree)		
normal_sample()		
if (2nd corner adjacent to tree)		
normal_sample()		
if (3rd corner adjacent to tree)		
normal_sample()		
}		
}		

color_updata() {		

```
if (color_binding=='01') {
        if (!visited)
            color_sample()
    }
    else if (color_binding =='10') {
        color_sample()
    }
    else if (color_binding == '11') {
        if (1st corner adjacent to tree)
            color_sample()
        if (2nd corner adjacent to tree)
            color_sample()
        if (3rd corner adjacent to tree)
            color_sample()
    }
}
```

```
texCoord_updata() {
    if (texCoord_binding=='01') {
        if (!visited)
            texCoord_sample()
    }
```

else if (texCoord_binding =='10') {		
texCoord_sample()		
}		
else if (texCoord_binding == '11') {		
if (1st corner adjacent to tree)		
texCoord_sample()		
if (2nd corner adjacent to tree)		
texCoord_sample()		
if (3rd corner adjacent to tree)		
texCoord_sample()		
}		
}		

[Effect of the Invention]

As described above, in a progressive 3-D mesh information coding/decoding method and apparatus according to the present invention, the apparatus has a data structure in which data is dealt with as mesh components (MCOM), so that division by components is possible even on a compressed bit stream. Thus, a decoding portion can easily reproduce components as soon as decoding by components is completed. Also, progressive image reproduction and capability of coping with transmission errors is better reinforced than in conventional 3-D mesh information coding methods. Furthermore, when a model is divided into mesh object layers (MOLs) or mesh components (MCOMs), the MOLs or MCOMs can be independently coded and decoded. Such a structure is simple, and can be realized so that redundant information can be easily removed as necessary.

What is claimed is:

1. A progressive 3-D mesh information coding method comprising the steps of:

(a) dividing a 3-D mesh into a plurality of mesh components;

(b) coding each of the plurality of mesh components; and

(c) multiplexing the plurality of coded mesh components into a compressed bit stream and transmitting the compressed bit stream.

2. The progressive 3-D mesh information coding method as claimed in claim 1, wherein each of the plurality of mesh components includes at least connectivity information, geometry information and photometry information which are necessary to reconstruct the coded mesh components themselves.

3. The progressive 3-D mesh information coding method as claimed in claim 1, wherein the step (a) comprises the substeps of:

(a1) extracting one or more mesh object layeres from a 3-D mesh; and

(a2) dividing one or more mesh object layeres each into a plurality of mesh components.

4. The progressive 3-D mesh information coding method as claimed in claim 1, wherein in the step (b), each of the plurality of mesh components is coded, and information generated while a mesh component is coded is reused in the process for coding a mesh component which has not yet been coded.

5. A progressive 3-D mesh information decoding method comprising the steps of:

(a) dividing a transmitted bit stream into a plurality of coded mesh components;

(b) decoding each of the plurality of coded mesh components; and

(c) reconstructing a 3-D mesh by synthesizing the plurality of decoded mesh components.

6. The progressive 3-D mesh information decoding method as claimed in claim 5, wherein the step (a) comprises the substeps of:

(a1) classifying the transmitted bit stream into one or more decoded mesh object layeres; and

(a2) dividing each of the one or more decoded mesh object layeres into a plurality of mesh components.

7. The progressive 3-D mesh information decoding method as claimed in claim 5, wherein in the step (b), each of the plurality of coded mesh components is decoded, and information generated while a mesh component is decoded is reused in the process for decoding a mesh component which has not yet been decoded.

8. A progressive 3-D mesh information coding/decoding method comprising the steps of:

(a) extracting one or more independent mesh object layers from a 3-D mesh;

(b) independently coding and transmitting the one or more mesh object layeres; and

(c) obtaining one or more independent mesh object layeres by decoding one or more of the independently coded and transmitted mesh object layeres.

9. The progressive 3-D mesh information coding/decoding method as claimed in claim 8, further comprising the step of (d) reconstructing the original 3-D mesh by collecting the independent mesh object layeres and removing redundant information, after step (c).

10. A progressive 3-D mesh information coding/decoding method comprising the steps of:

(a) extracting one or more mesh object layers from a 3-D mesh and dividing each of the mesh object layers into a plurality of independent mesh components;

(b) independently coding and transmitting the plurality of mesh components for each of the one or more mesh object layers; and

(c) obtaining a plurality of independent mesh components by decoding the plurality of independently coded and transmitted mesh components.

11. The progressive 3-D mesh information coding/decoding method as claimed in claim 10, further comprising the step of (d) reconstructing the original 3-D mesh by collecting the independent mesh components and removing redundant information between adjacent mesh components, after step (c).

12. A progressive 3-D mesh information coding apparatus comprising:

a 3-D data analyzer for receiving a 3-D mesh and reconstructing the input 3-D mesh into a plurality of mesh components;

a plurality of component coders for coding the plurality of mesh components; and

a multiplexer for multiplexing the plurality of coded mesh components into a compressed bit stream.

13. The progressive 3-D mesh information coding apparatus as claimed in claim 12, wherein the 3-D data analyzer comprises:

a 3-D mesh object layer analyzer for dividing the input 3-D mesh into one or more mesh object layeres; and

a plurality of mesh component analyzers for again dividing each of one or more mesh object layeres into a plurality of mesh components.

14. The progressive 3-D mesh information coding apparatus as claimed in claim 12, wherein when the plurality of component coders code the plurality of mesh components, a component coder which has not yet performed coding performs coding using coding information generated in a component coder which has already performed coding.

15. A progressive 3-D mesh information decoding apparatus for progressively decoding a bit stream decoded by and transmitted from a progressive 3-D mesh information coding apparatus, comprising:

a demultiplexer for dividing the transmitted bit stream into a plurality of coded mesh components;

a plurality of component decoders for decoding the plurality of coded mesh components; and

a 3-D data synthesizer for synthesizing the plurality of decoded mesh components to reconstruct a 3-D mesh.

16. The progressive 3-D mesh information decoding apparatus as claimed in claim 15, wherein when the plurality of component decoders decode the plurality of mesh components, a component decoder which has not yet performed decoding performs decoding using decoding information generated in a component decoder which has already performed decoding.

17. A progressive 3-D mesh information coding/decoding apparatus comprising:

a 3-D mesh object layer analyzer for receiving a 3-D mesh and dividing an input 3-D mesh into one or more independent mesh object layeres;

one or more mesh object layer coders for independently coding and transmitting one or more mesh object layeres; and

one or more mesh object layer decoders for decoding one or more independent mesh object layeres which are independently coded and transmitted, to obtain one or more independent mesh object layeres.

18. An independent and progressive 3-D mesh information coding/decoding apparatus as claimed in claim 17, further comprising a 3-D mesh object layer synthesizer for synthesizing one or more independent mesh object layeres and removing redundant information to reconstruct the original 3-D mesh.

19. A progressive 3-D mesh information coding/decoding apparatus comprising:

a 3-D mesh object layer analyzer for receiving a 3-D mesh, dividing an input 3-D mesh into one or more mesh object layeres, and again dividing each mesh object layer into a plurality of independent mesh components;

a plurality of mesh component coders for independently coding and transmitting the plurality of mesh object layers; and

a plurality of mesh component decoders for decoding the plurality of mesh components which are independently coded and transmitted, to obtain a plurality of independent mesh components.

20. An independent and progressive 3-D mesh information coding/decoding apparatus as claimed in claim 19, further comprising a 3-D step data synthesizer for synthesizing the plurality of independent mesh components and removing redundant information between adjacent mesh components to reconstruct the original 3-D mesh.

FIG. 1

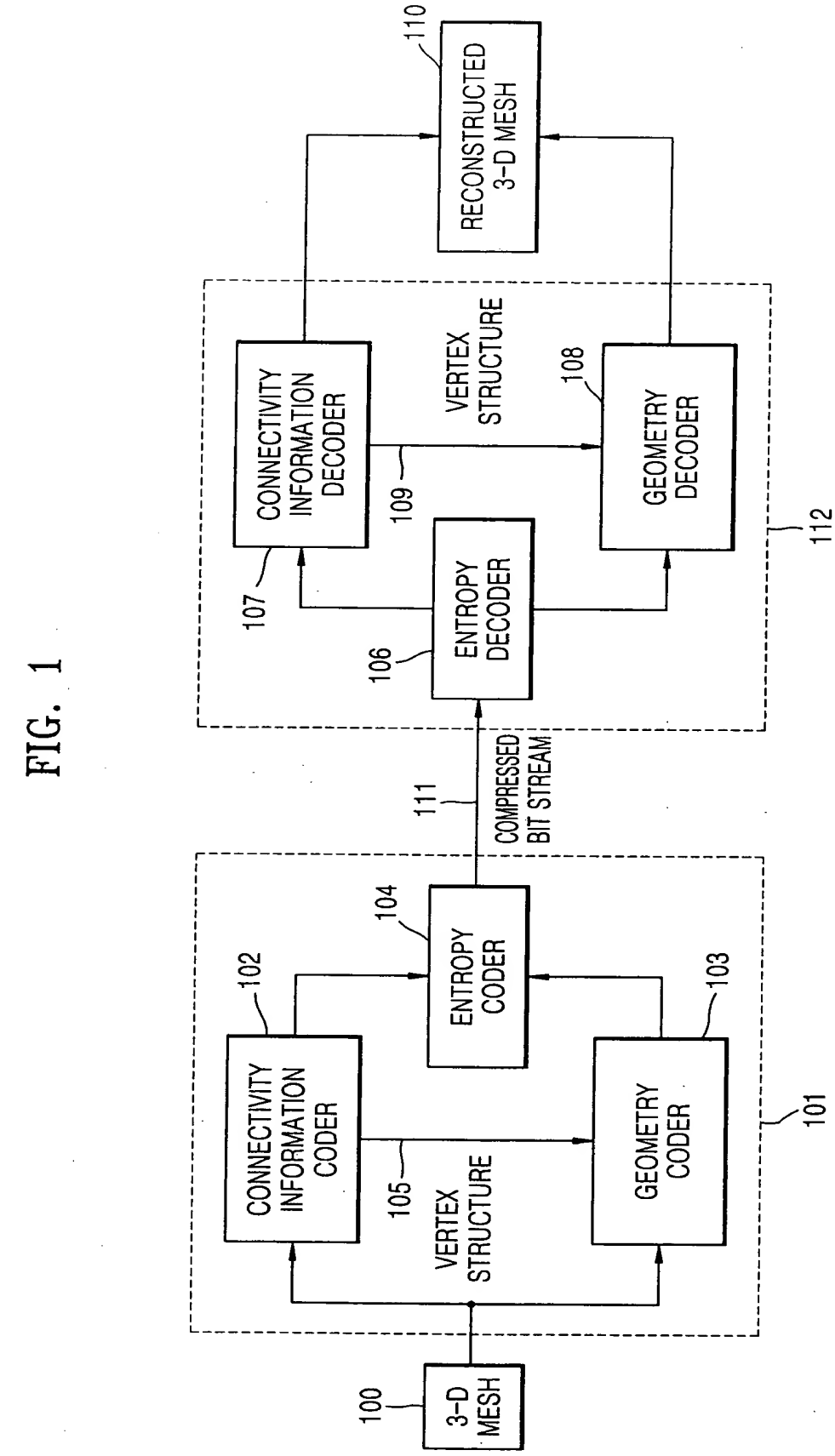


FIG. 2

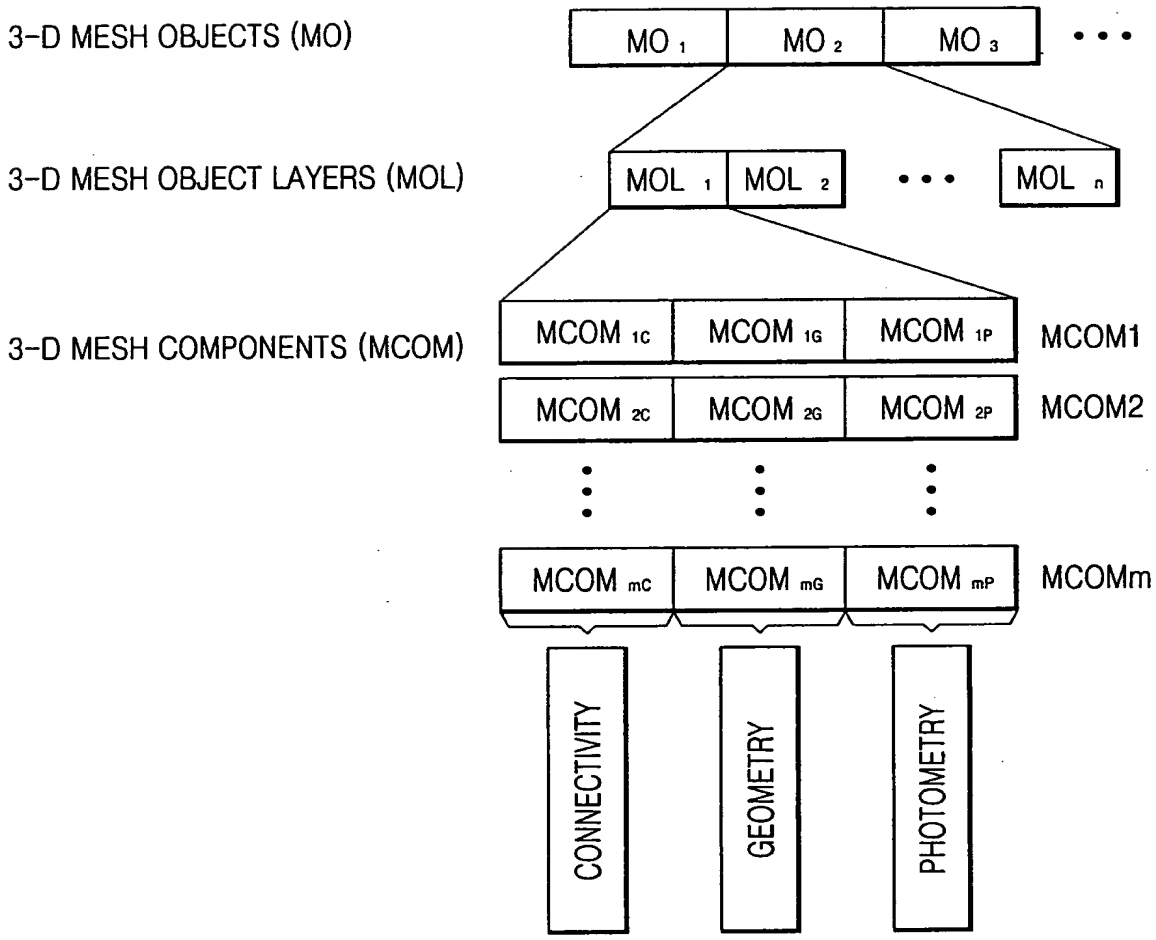

3-D MESH OBJECTS (MO)
MO 1
MO 2
MO 3
3-D MESH OBJECT LAYERS (MOL)
MOL 1
MOL 2
MOL n
3-D MESH COMPONENTS (MCOM)
MCOM 1C
MCOM 1G
MCOM 1P
MCOM1
MCOM 2C
MCOM 2G
MCOM 2P
MCOM2
MCOM mC
MCOM mG
MCOM mP
MCOMm
CONNECTIVITY
GEOMETRY
PHOTOMETRY

FIG. 3

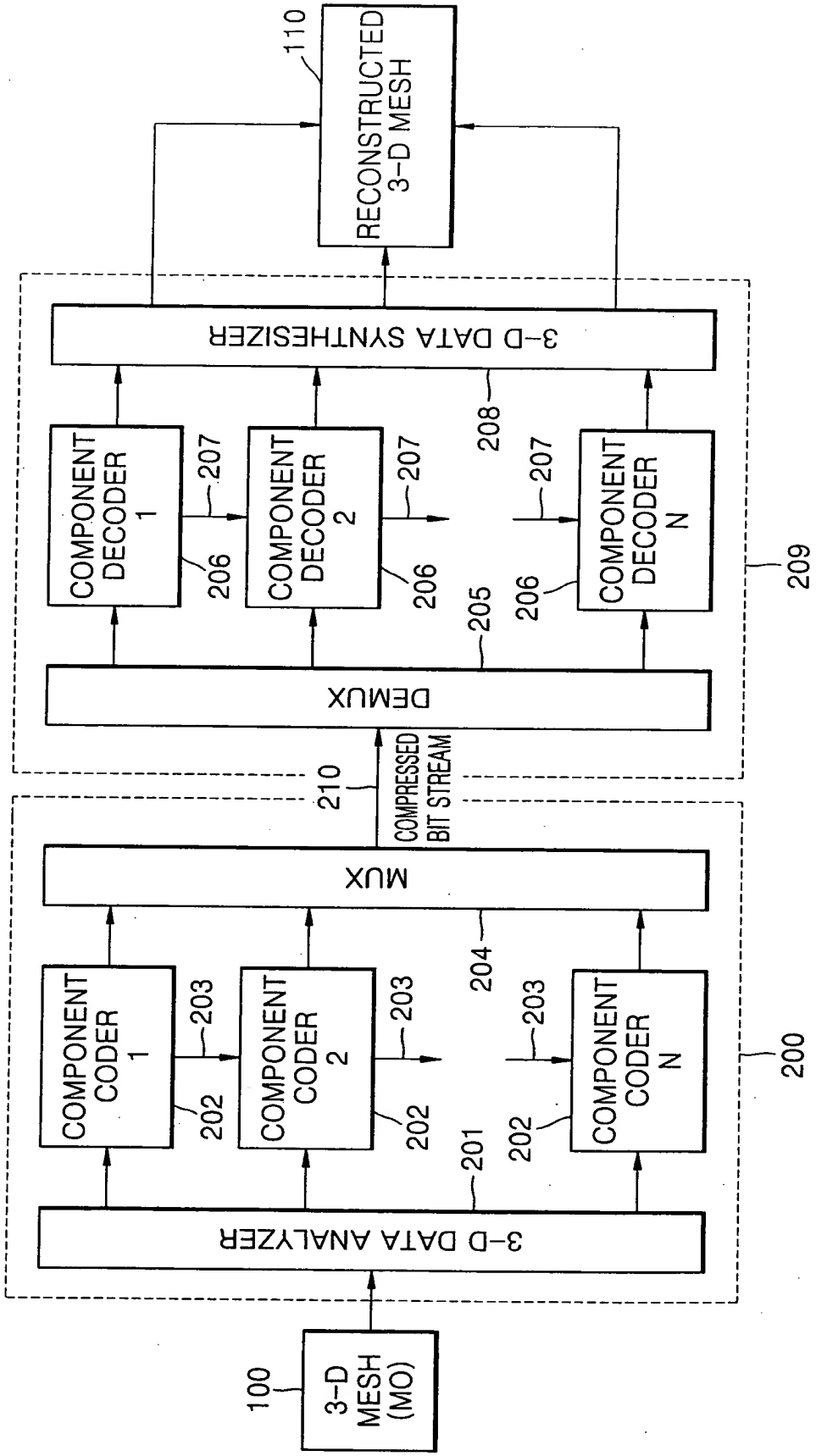

100
3-D MESH (MO)
3-D DATA ANALYZER
201
202
COMPONENT CODER 1
203
COMPONENT CODER 2
204
COMPONENT CODER N
MUX
200
210
COMPRESSED BIT STREAM
DEMUX
205
206
COMPONENT DECODER 1
207
COMPONENT DECODER 2
208
COMPONENT DECODER N
209
3-D DATA SYNTHESIZER
110
RECONSTRUCTED 3-D MESH

FIG. 4

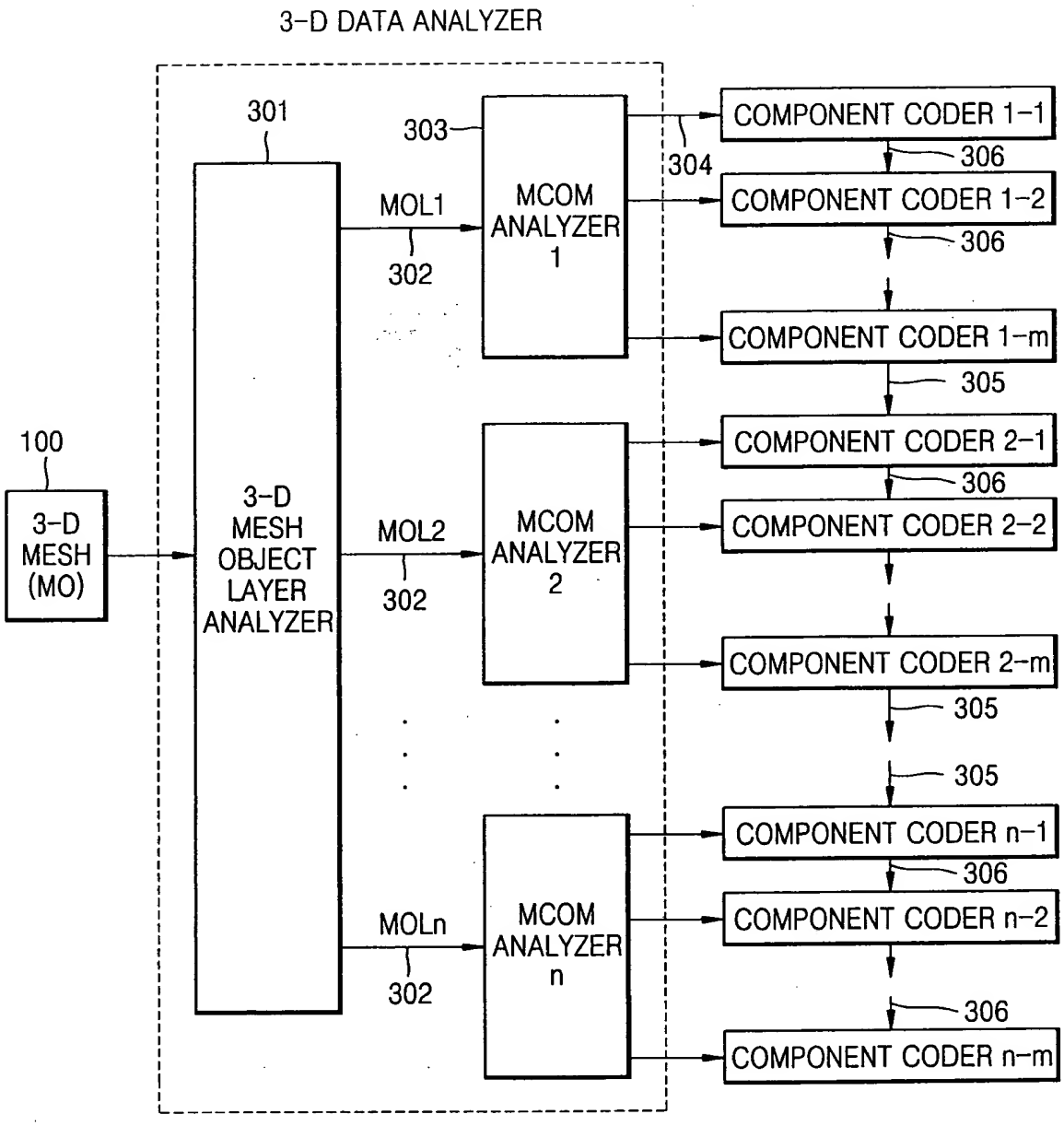

3-D DATA ANALYZER
100
3-D MESH (MO)
301
3-D MESH OBJECT LAYER ANALYZER
303
MOL1
302
MCOM ANALYZER 1
MOL2
302
MCOM ANALYZER 2
MOLn
302
MCOM ANALYZER n
304
COMPONENT CODER 1-1
306
COMPONENT CODER 1-2
306
COMPONENT CODER 1-m
305
COMPONENT CODER 2-1
306
COMPONENT CODER 2-2
COMPONENT CODER 2-m
305
305
COMPONENT CODER n-1
306
COMPONENT CODER n-2
306
COMPONENT CODER n-m

FIG. 5

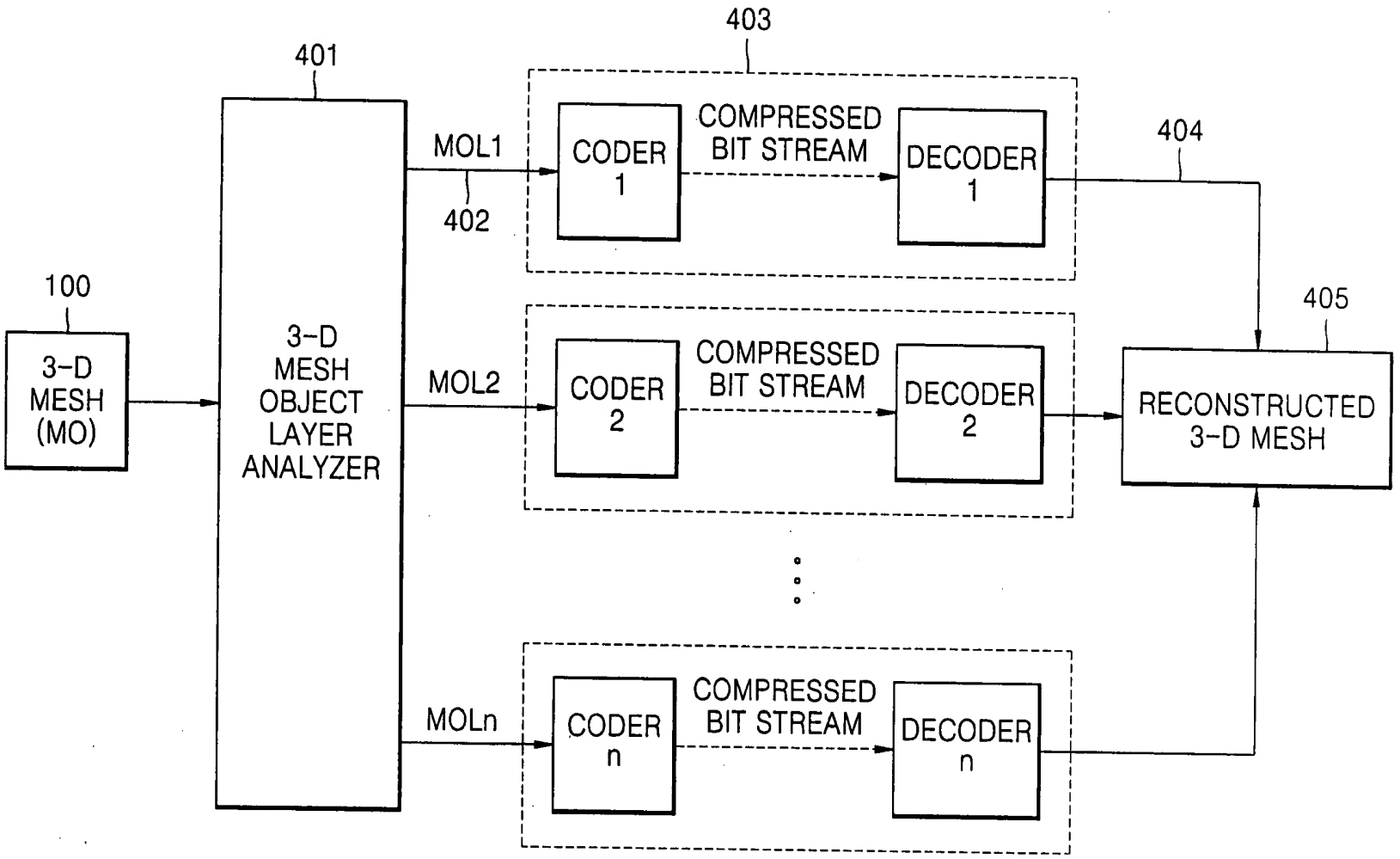

403
401
MOL1
CODER 1
COMPRESSED BIT STREAM
DECODER 1
404
402
100
405
3-D MESH (MO)
3-D MESH OBJECT LAYER ANALYZER
MOL2
CODER 2
COMPRESSED BIT STREAM
DECODER 2
RECONSTRUCTED 3-D MESH
MOLn
CODER n
COMPRESSED BIT STREAM
DECODER n

FIG. 6

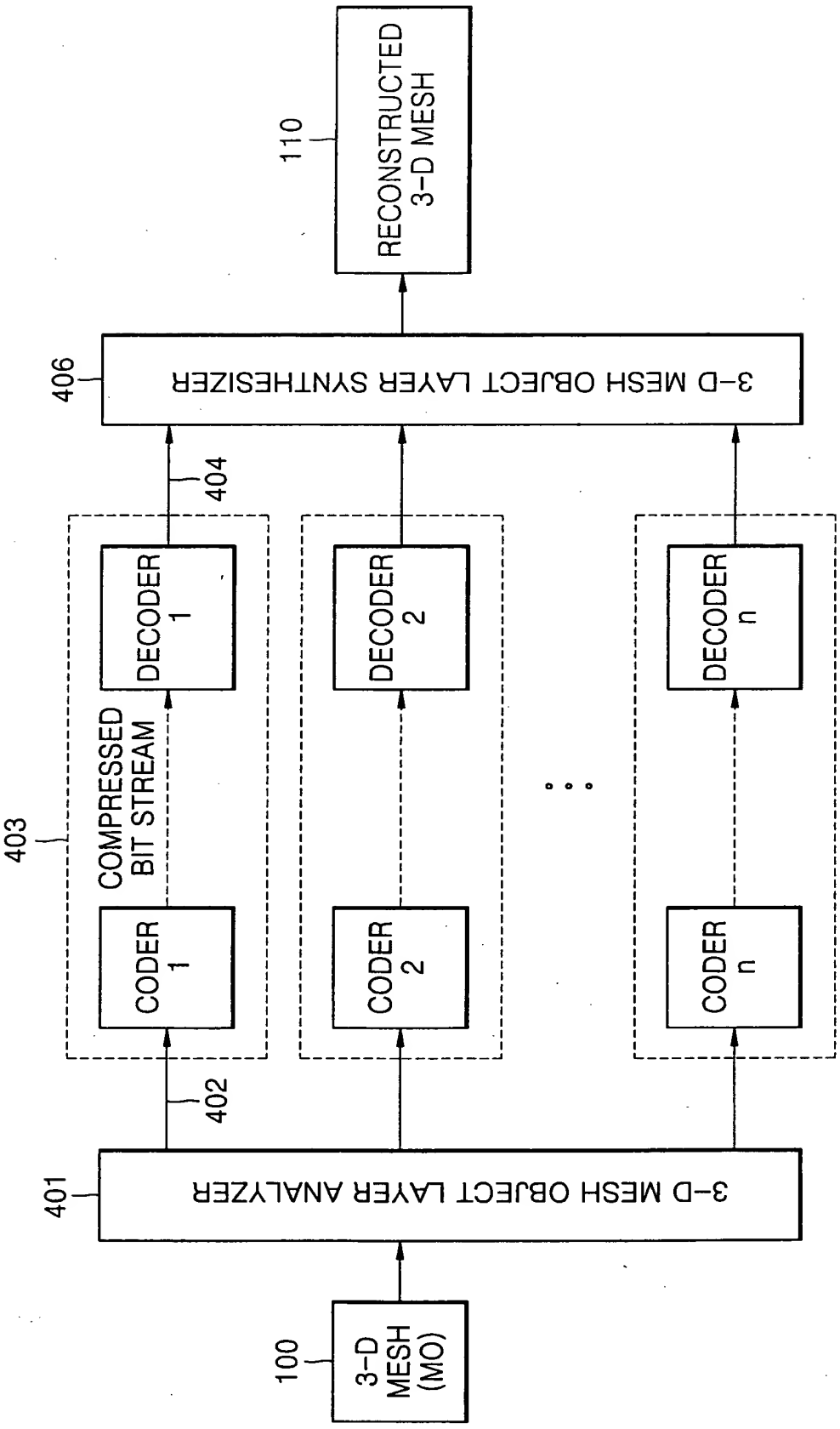

100
3-D MESH (MO)
401
3-D MESH OBJECT LAYER ANALYZER
402
403
COMPRESSED BIT STREAM
CODER 1
CODER 2
CODER n
404
DECODER 1
DECODER 2
DECODER n
406
3-D MESH OBJECT LAYER SYNTHESIZER
110
RECONSTRUCTED 3-D MESH

FIG. 7

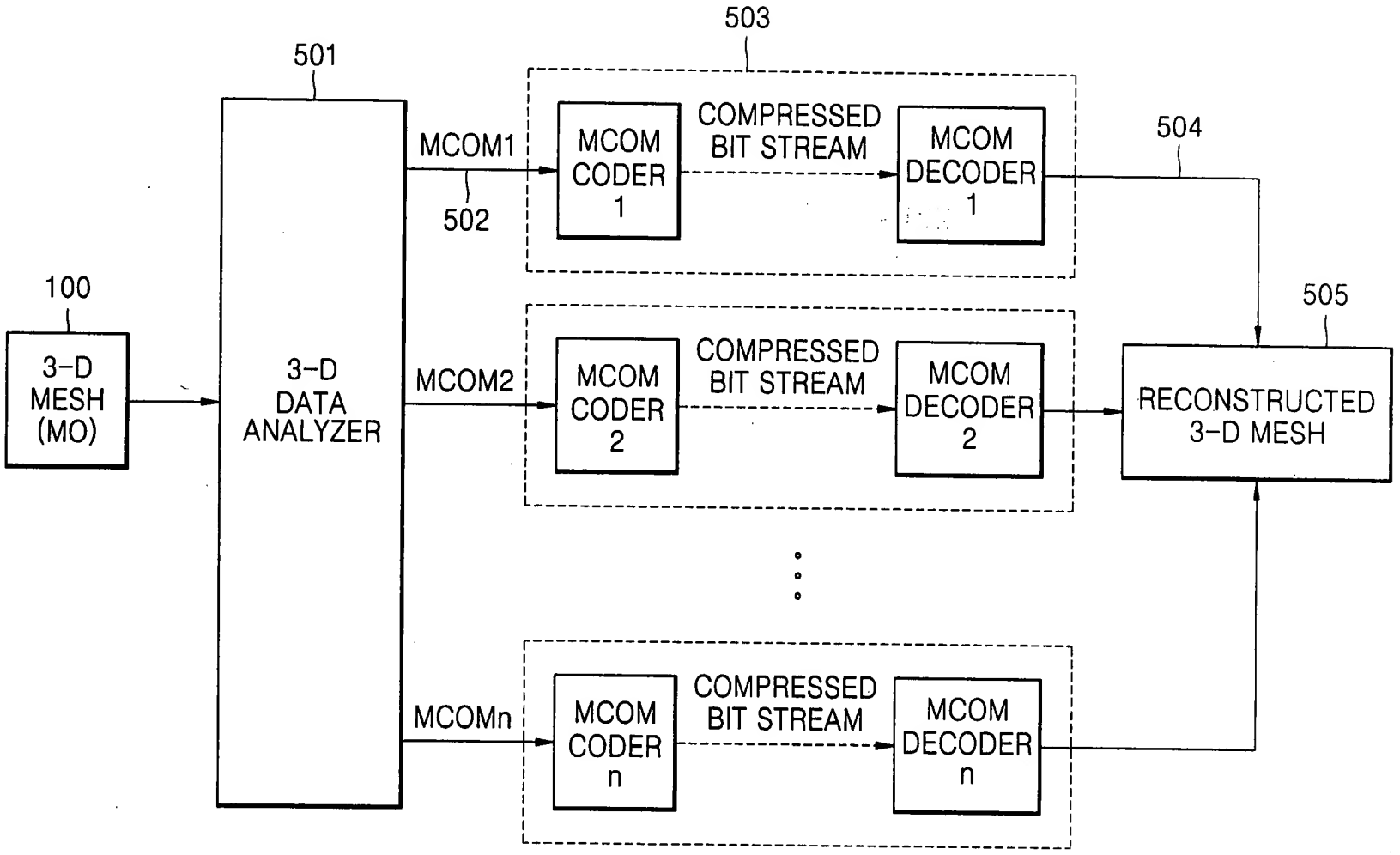

503
501
504
MCOM1
MCOM CODER 1
COMPRESSED BIT STREAM
MCOM DECODER 1
502
100
505
3-D MESH (MO)
3-D DATA ANALYZER
MCOM2
MCOM CODER 2
COMPRESSED BIT STREAM
MCOM DECODER 2
RECONSTRUCTED 3-D MESH
MCOMn
MCOM CODER n
COMPRESSED BIT STREAM
MCOM DECODER n

FIG. 8

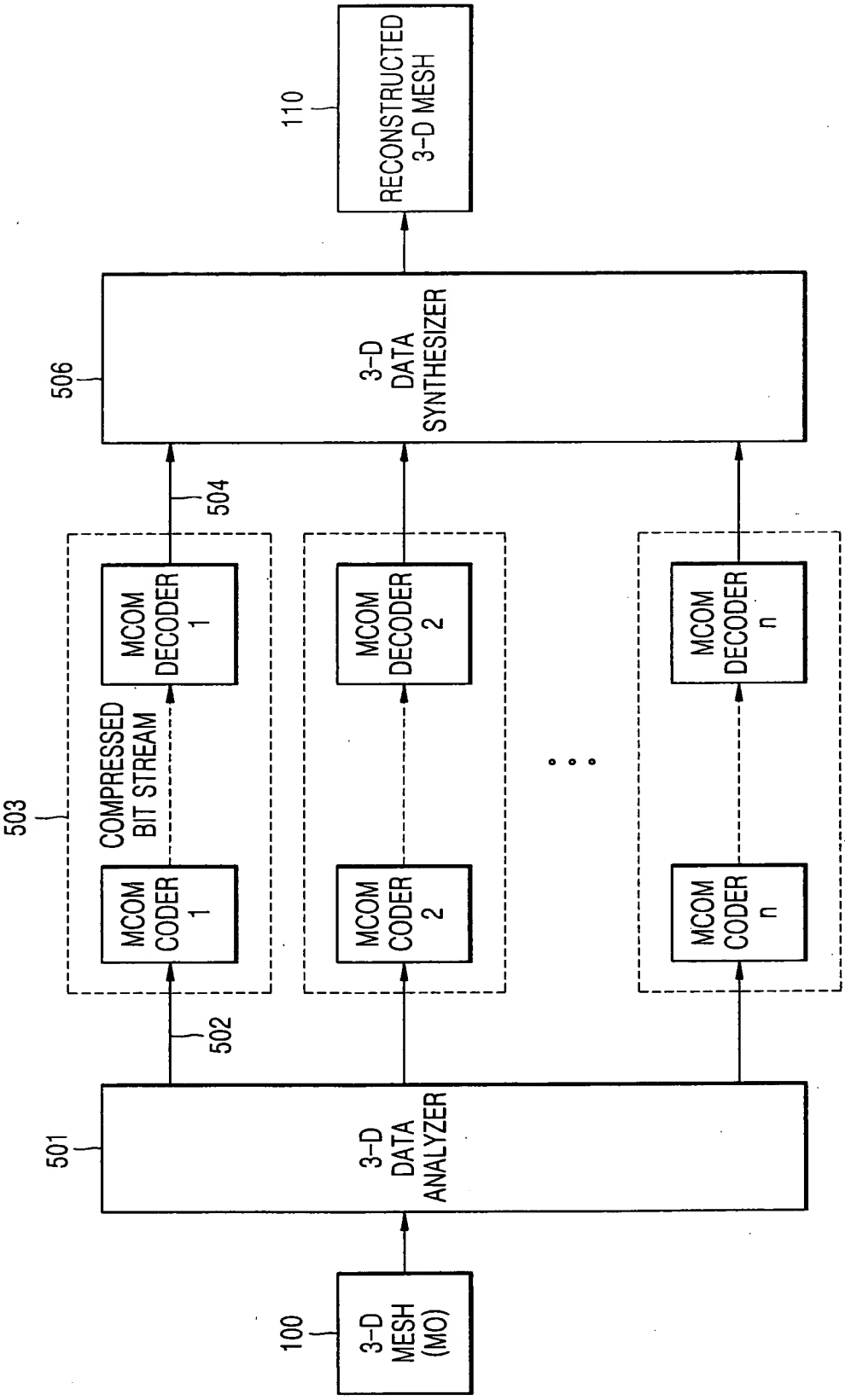

100
3-D MESH (MO)
501
3-D DATA ANALYZER
502
503
COMPRESSED BIT STREAM
MCOM CODER 1
MCOM CODER 2
MCOM CODER n
MCOM DECODER 1
MCOM DECODER 2
MCOM DECODER n
504
506
3-D DATA SYNTHESIZER
110
RECONSTRUCTED 3-D MESH

FIG. 9

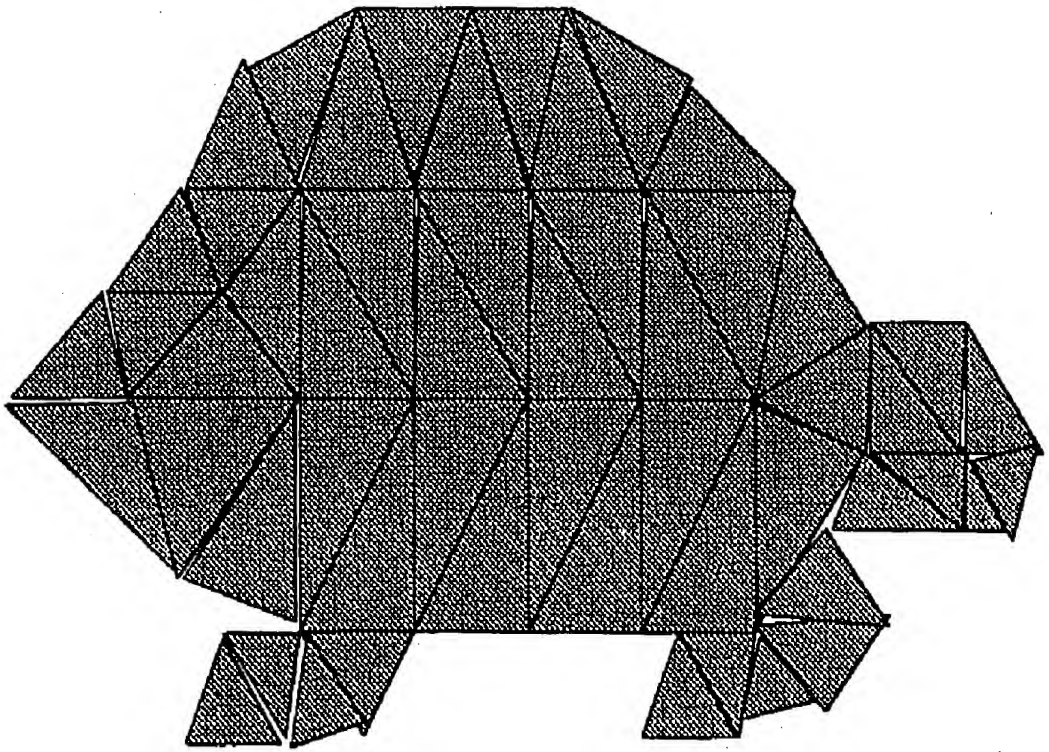


FIG. 10

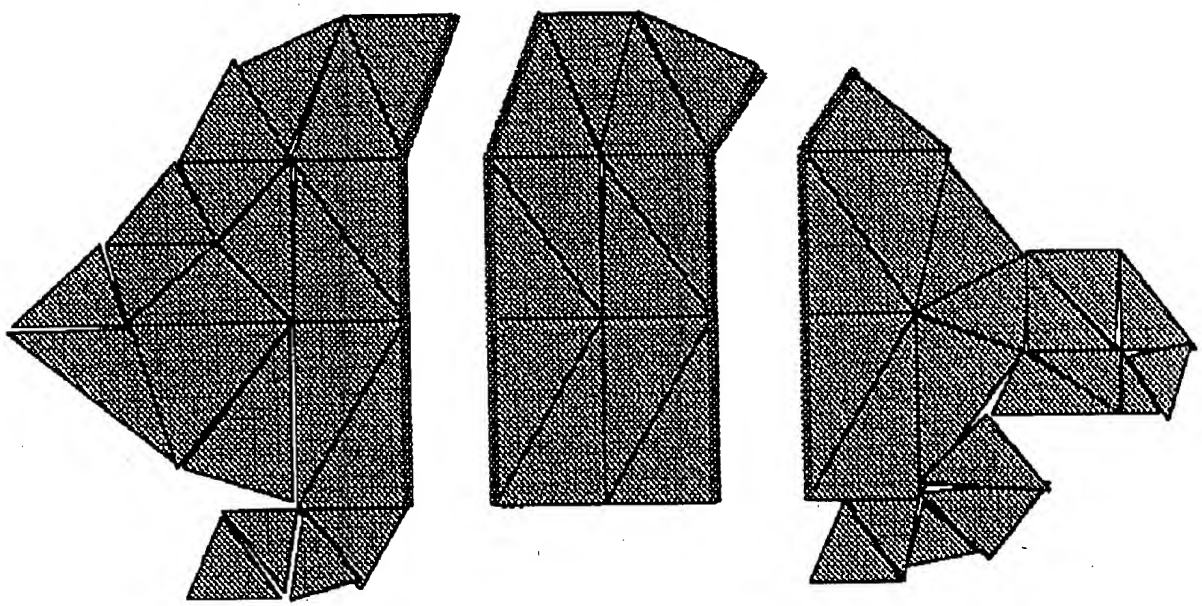


FIG. 11

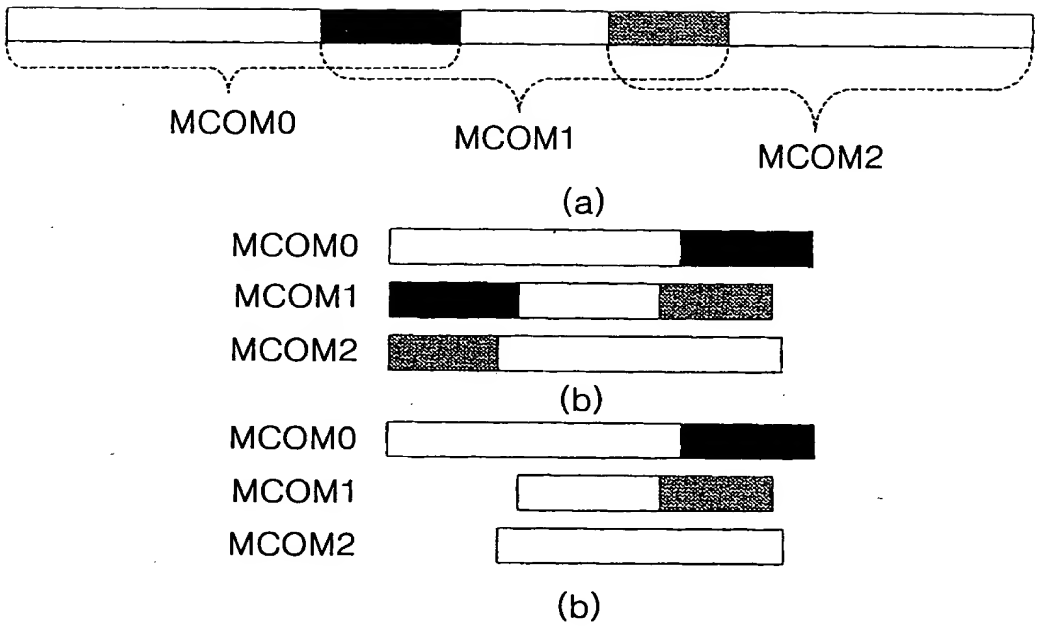
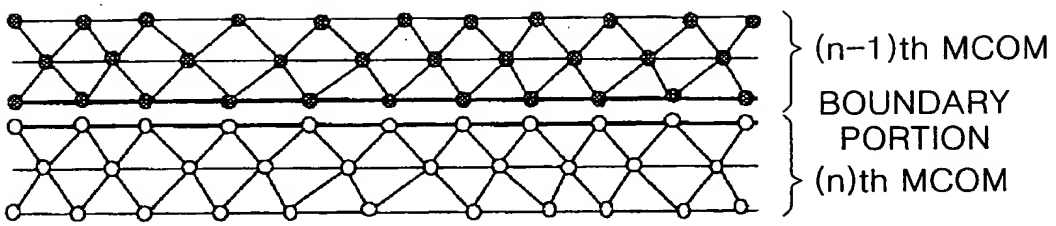
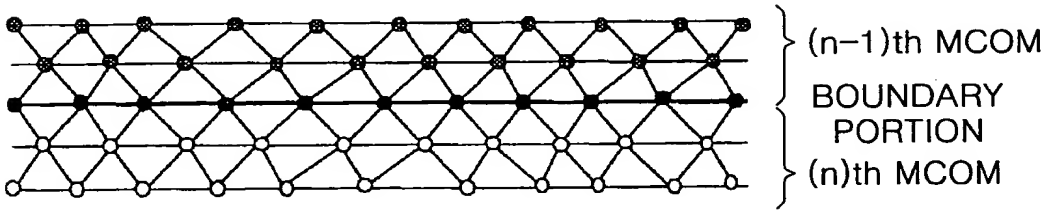


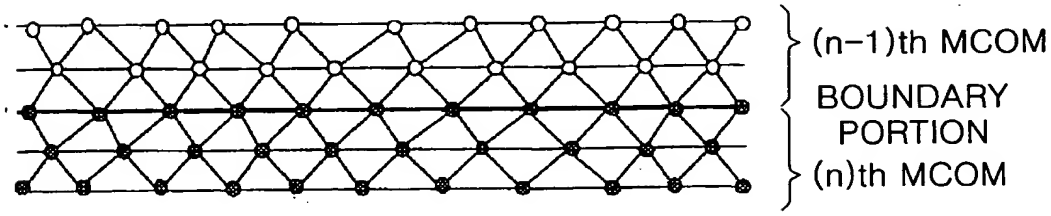
FIG. 12


(n-1)th MCOM
BOUNDARY PORTION
(n)th MCOM

(a)


(n-1)th MCOM
BOUNDARY PORTION
(n)th MCOM

(b)


(n-1)th MCOM
BOUNDARY PORTION
(n)th MCOM

(c)

08/27/98

	Class	Subclass

ISSUE CLASSIFICATION

SCANNED3

PROVISIONAL APPLICATION NUMBER

BEST AVAILABLE COPY

Form PTO-1625-13
(Rev. 05-03)

SCAN 3

SCAN 5 A620 C.Dm

(FACE)

PATENT APPLICATION SERIAL NO. 60/098150

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

09/02/1998 POLLEN 00000063 122225 60098150
01 FC:114 150.00 CH

PTO-1556
(5/87)

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
60/098,150 PROVISIONAL	08/27/98		0000	1-1-27-1

APPLICANT

CHANDRAJIT BAJAJ, AUSTIN, TX; VALERIO PASCUCCI, AUSTIN, TX; ERIC DAVID PETAJAN, WATCHUNG, NJ; GUOZHONG ZHUANG, AUSTIN, TX.

CONTINUING DOMESTIC DATA********
VERIFIED

_____

371 (NAT'L STAGE) DATA********
VERIFIED

_____

FOREIGN APPLICATIONS********
VERIFIED

_____

FOREIGN FILING LICENSE GRANTED 09/25/98

Foreign Priority claimed 35 USC 119 (a-d) conditions met	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
☐yes ☐no ☐yes ☐no ☐Met after Allowance Verified and Acknowledged _____ Examiner's Initials _____ Initials	TX	0		

ADDRESS

LUENT TECHNOLOGIES INC
600 MONTAIN AVENUE
P O BOX 636
MURRAY HILL NJ 07974-0636

TITLE

COMPRESSION AND PROGRESSIVE TRANSMISSION OF ARBITRARY POLYGONAL

FILING FEE RECEIVED	FEES: Authority has been given in Paper	
$150	No. _____ to charge/credit DEPOSIT ACCOUNT NO. _____ for the following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

A/ERDV

"Express Mail" mailing label number EE718068712US AUG 27 1998
Date of Deposit
I hereby certify that this APPLICATION is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks Washington, D. C. 20231
CRIS F. THOMAS
(Printed name of person mailing paper or fee)
AUG 27 1998
(Signature of person mailing paper or fee)

IN THE UNITED STATES
ENT AND TRADEMARK OFFICE

08/27/98 Lucent

PROVISIONAL APPLICATION

Chandrajit Bajaj
Valerio Pascucci
Eric David Petajan
Guozhong Zhuang

Case 1-1-27-1

Title Compression And Progressive Transmission Of Arbitrary Polygonal Meshes

JC541 U.S. PTO 60/098150 08/27/98

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

PROVISIONAL APPLICATION COVER SHEET

SIR:

This is a request to file a **Provisional Application** under **37 CFR 1.53 (c).**

[✓] 97 pages in Specification
[ ] 0 sheet(s) of Drawing(s)
[ ] The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.
[ ] The name of the U.S. Government agency and the Government contract number are:

Inventor(s): (full name and address)

Chandrajit Bajaj, 2632 Creeks Edge Parkway, Austin, TX 78733 USA
Valerio Pascucci, 400 West Anderson Lane, Austin, TX 78752 USA
Eric David Petajan, 22 Maple Street, Watchung, NJ 07060 USA
Guozhong Zhuang, 2501 Lake Austin Blvd., Apt. K104, Austin, TX 78703 USA

The Commissioner is hereby authorized to charge **Lucent Technologies Deposit Account No. 12-2325** the amount of **$150.00** to cover the cost of the filing fee. In the event of any non-payment or improper payment of a required fee, the Commissioner is authorized to charge or to credit Account No. 12-2325 as required to correct the error.

Please address all correspondence to **Docket Administrator (Room 3C-512), Lucent Technologies Inc., 600 Mountain Avenue, P. O. Box 636, Murray Hill, New Jersey 07974-0636.** However, telephone calls should be made to me at 908-582-5998.

Respectfully,

Kenneth M. Brown
Attorney for Applicant(s)
Reg. No. 37590

Date: 8/27/98

Lucent Technologies Inc.
600 Mountain Avenue (Room 3C-512)
P. O. Box 636
Murray Hill, New Jersey 07974-0636

PT 974 10/97

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

"Express Mail" mailing label number EE718068712US AUG 27 1998
Date of Deposit
I hereby certify that this APPLICATION is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks Washington, D. C. 20231.
IRIS F. THOMAS
(Printed name of person mailing paper or fee)
AUG 27 1998
(Signature of person mailing paper or fee)

PROVISIONAL APPLICATION

Chandrajit Bajaj
Valerio Pascucci
Eric David Petajan
Guozhong Zhuang

Case 1-1-27-1

Title Compression And Progressive Transmission Of Arbitrary Polygonal Meshes

JC841 U.S. PTO
60/099150
08/27/98

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

PROVISIONAL APPLICATION COVER SHEET

SIR:

This is a request to file a **Provisional Application** under **37 CFR 1.53 (c).**

[✓] 97 pages in Specification
[ ] 0 sheet(s) of Drawing(s)
[ ] The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.
[ ] The name of the U.S. Government agency and the Government contract number are:

Inventor(s): (full name and address)

Chandrajit Bajaj, 2632 Creeks Edge Parkway, Austin, TX 78733 USA
Valerio Pascucci, 400 West Anderson Lane, Austin, TX 78752 USA
Eric David Petajan, 22 Maple Street, Watchung, NJ 07060 USA
Guozhong Zhuang, 2501 Lake Austin Blvd., Apt. K104, Austin, TX 78703 USA

The Commissioner is hereby authorized to charge **Lucent Technologies Deposit Account No. 12-2325** the amount of **$150.00** to cover the cost of the filing fee. In the event of any non-payment or improper payment of a required fee, the Commissioner is authorized to charge or to credit Account No. 12-2325 as required to correct the error.

Please address all correspondence to **Docket Administrator (Room 3C-512), Lucent Technologies Inc., 600 Mountain Avenue, P. O. Box 636, Murray Hill, New Jersey 07974-0636.** However, telephone calls should be made to me at 908-582-5998.

Respectfully,

Kenneth M. Brown
Attorney for Applicant(s)
Reg. No. 37590

Date: 8/27/98

Lucent Technologies Inc.
600 Mountain Avenue (Room 3C-512)
P. O. Box 636
Murray Hill, New Jersey 07974-0636

PT 974 10/97

Compression and Progressive Transmission of Arbitrary Polygonal Meshes

Category: Computing Methodologies

Abstract

We present a topological layering scheme coupled with vector quantization for compressing both the topology (connectivity) and geometry (vertex coordinates) of arbitrary polygon meshes. The polygon mesh surface could be open or closed, non-manifold, and with multiple holes (arbitrary genus). The layered topological decomposition makes the compression and connectivity encoding as efficient as that for ribbon surfaces or triangle strips. The vector quantization and geometry encoding of the vertex coordinates is done with novel error/distortion control parameters and allows for progressive transmission of the encoded information. The separation of topology and geometry encoding permits all combinations of lossy or lossless topology and lossy or lossless geometry.

Keywords: Polygon Meshes, Geometry Compression, Connectivity Encoding, Vector Quantization

1 Introduction

Interactively creating, manipulating and transmitting 3D geometry, especially in remote networked environments has become increasingly important. A polygon (or triangular) mesh

is the primary geometric model representation in computer graphics. In most cases, accurate polygon mesh approximation of a surface with sharp features (holes, highly varying curvatures) requires extremely large number of triangles. For example, current generation computer models of military vehicles, aircrafts or submarines often consist of a few million triangles. Direct manipulation or transmission of these meshed objects in a naive way will not be practical ([15]) given current improvement trends in communication and graphics hardware. Elegant algorithms are needed to significantly reduce both data access and rendering times. Such software acceleration techniques include dynamic simplification based on visibility [8], [10], [20], reduced resolution tradeoffs with accuracy [16], [18], [12] and compression and encoding [3], [17].

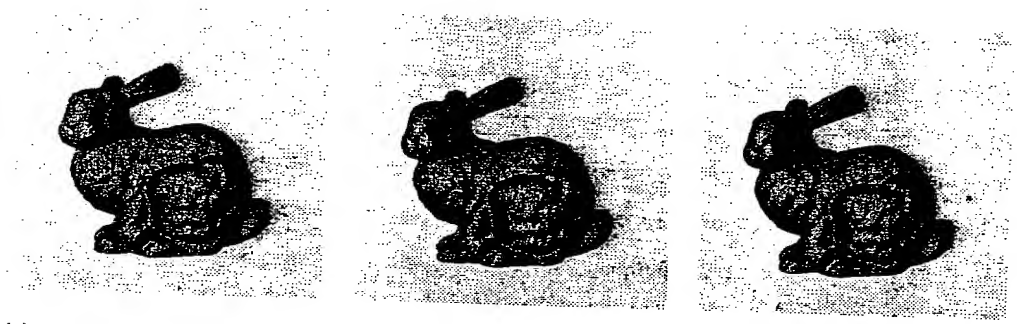
Polygon mesh compression is a more general technique which includes geometric compression (lossy or lossless encoding of numeric data such as vertex coordinates) and topology compression (lossy or lossless encoding of symbolic data such connectivity and adjacency). Direct polygon mesh simplification schemes such as [16, 14] are a form of "lossy" topology encoding or compression. Under the premise of preserving a prescribed geometric or visual accuracy, polygon meshes compression methods provide ways to minimize the data storage space with encoded (and sometimes simplified) representations of the original models.

Previous Related Work Overview By representing polygon mesh connectivity as generalized triangle strips, Deering perhaps is the first to introduce to the computer graphics community the concept of geometry compression concerns about efficient encoding of both structure data (connectivity) and attribute data (geometry). The main objective of geometric compression and encoding is to exploit the redundancy contained in the data. The trick of

Deering's method [3] lies in its ability to compactly express a planar graph via stack operators so that a large number of the mesh vertices can be re-utilized. The topological surgery paper by Taubin and Rossignac [17] is quite similar. A polygon mesh is first decomposed into triangle strips by use of a vertex spanning tree. Encoding of these strips can be guided by following the dual triangle tree in a depth-first-search manner. Geometry and connectivity encoding can be processed separately. While producing excellent compression that algorithm has limitations when dealing with closed meshes of arbitrary genus as well as in producing progressive transmission.

Popovic and Hoppe[14] propose a lossy compression (simplification) technique which is progressively transmittable. They express the original simplicial mesh at different resolutions through successive edge collapse and merge operations [8]. While not an efficient encoding method, its fundamental contribution is the progressive transmission of simplified mesh connectivity. Their method is elegantly improved [11] by integrating the structure bit stream and the attribute stream under some optimizing criteria. Chow [2] presents an optimized compression technique which decomposes a given mesh into generalized triangle meshes as well as incorporates a geometry and topology encoding scheme. An obvious shortcoming of his method is the large number of bits used in encoding the mesh connectivity ($\log n + 9$ per vertex).

Zorin, Schroder and Sweldens use subdivision to connect and unify patches and polygonal meshes in order to produce a tool to do mesh manipulation. Algorithms are designed for interactive multiresolution editing of meshes of arbitrary topology. They also claim that compression and progressive transmission are possible future research objectives because of the multiresolution transform.



(a) compressed (70,664 bytes) (b) compressed(81,882 bytes) (c) original (3,415,624 bytes)

Figure 1: The original bunny mesh has 34,835 vertices and 69,473 faces.

Main Results Our topology and geometry compression and encoding scheme (detailed in sections 2 and 3, see Figure 53), while similar in spirit to the methods of Deering, Taubin et. al., and Chow, is superior in the following aspect. It achieves higher compression ratios since an improved geometry encoding method is taken. It cherishes the capability of handling arbitrary polygon meshes (high genus,non-manifold) by taking a quite natural and fancy connectivity encoding scheme. No vertex spanning tree is needed to stitch the boundary edges of the strips to recover the original connectivity. Progressive transmission can be performed in two directions: either the bit transmission of the encoded geometry with error/distortion control parameters, or connectivity decomposition which allow incremental transmission. Empirical results are presented in section 4.

2 Compression and Encoding Techniques

In this section, we first describe how to express the connectivity information by a special kind of triangle strips which are created by the existing layered decomposition method. Ba-

sically, all vertices are grouped into separate layer. And each vertex layer is further divided into a set of contours and possibly some isolated vertices. Most triangles triangles are put into triangles strips which can be efficiently encoded. Special triangles will be treated specially. Finally, an improved geometry compression and encoding technique is adopted. Details are provided in the following subsections so that the reader can easily build his own implementation.

The concept of layered decomposition (layering) of a mesh is not new. Taubin and Rossignac first use it to build vertex spanning tree/forest and triangle tree/forest which are the central part of their geometric compression and encoding algorithms ([17]). Better compression ratios are produced by taking this approach.

The following is a slight generalized definition of layered decomposition. The notations of vertex layer and triangle layer used in [17] are still valid. A set of vertices are first chosen to make up of the first vertex layer. Usually we these vertices form a contour or polyline. The k^{th} triangle layer is composed of all the triangles, each of which has at least one vertex belonging to the k^{th} vertex layer. The $(k+1)^{th}$ vertex layer is then defined as the set of vertices in which each vertex appears at least once in the index set of all the $k-th$ layer triangles but not belongs to any of the first k vertex layers.

Organizing vertices and triangles in this way has at least two advantages. First, the vertex layer is mostly composed of several contours (3D polygons) or polyline (also called open contours). The spatial coherency in such expression can be used for efficient geometric compression and encoding. Second, the triangle layer can form a set of general triangle strips (see precise definition below). The strips can be efficiently encoded by the bit march approach which has been used in [17].

Furthermore, our algorithm based on the layering scheme is very efficient on a set of

meshes, the tiling surfaces. A tiling surface is a triangle mesh which is reconstructed from planar cross sections for from volumetric images ([1], [13], [4]). If we choose all the vertices in the bottom cross section as our first vertex layer, then vertices in any later vertex layer come from a single cross-section. And a triangle layer is composed of all tiling triangles between two adjacent cross sections. Without loss of generality, assume that all cross sections are parallel to the XOY plane. Then all vertices in any vertex layer have a same z value. So an obvious saving in geometric encoding is to encode z value once per layer. The original idea of our compression scheme comes from this. But we generalized it to deal with meshes of arbitrary topology.

Before describing our approach, we make it clear first that although our approach will uses layering scheme to organize connectivity information, it is different from any existing connectivity scheme. It encodes the geometry and connectivity in quite a natural way. It is both efficient and robust. Moreover, it can be applied to meshes of any manifold. These features will be explained in the the left part of this paper.

Figure 2(a) shows the progressive layering construction starting from a single vertex while Figure 2(b) shows the case of the layer which is a contour.

In both cases one performs the construction simply by grouping in layer i all (and only) the vertices that are edge-connected to a vertex in layer $i-1$. The choice on layer 0 remains a free parameter, that in theory may influence the resulting layering structure. From our experimental results, however, we have found that the quality of the resulting layering (like average number of layers) does not substantially change due to the initial choice.

It is important to note that the set of vertices in a single layer may not be connected by edges of the mesh. In ths case we say that a branching has occurred so that a layer of vertices is decomposed into a number of contours. Figure 2(c) shows a simple case where the initial

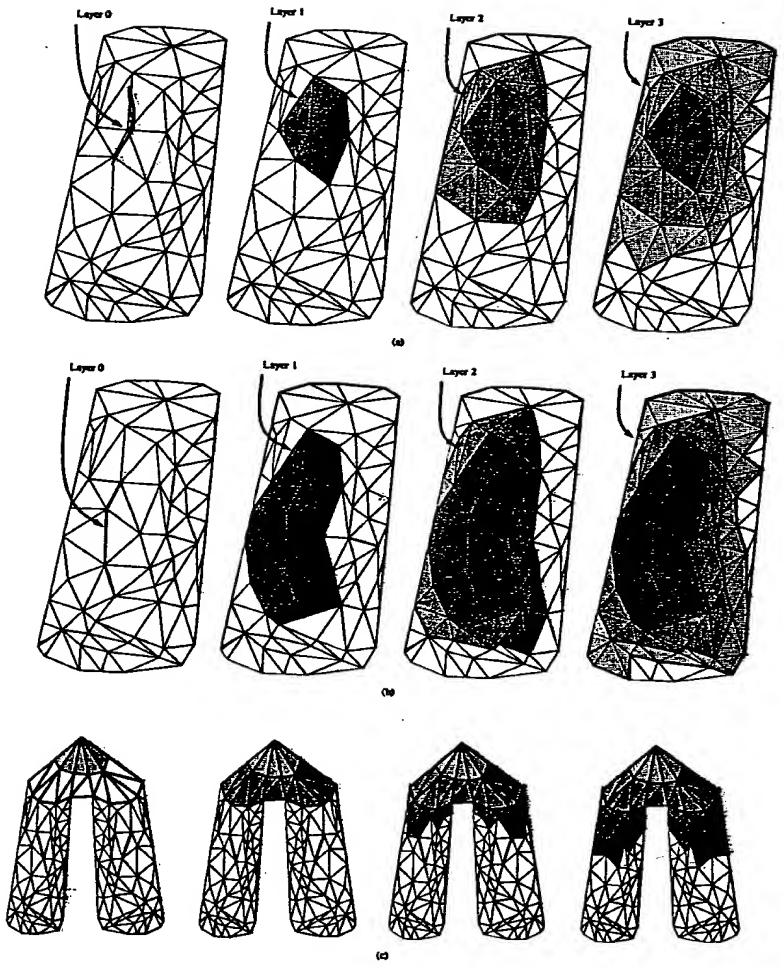


Figure 2: (a) Layer decomposition with a single starting point. (b) Layer decomposition which first layer is a path. (c) Layer decomposition with branching.

layer at a certain stage branches into two contours.

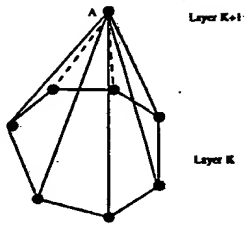


Figure 3: Isolated vertex

An edge that connects two same layer vertices is called in-between edge. As we mentioned above, vertices in a layer are grouped into contours (open or closed). Basically, these vertices are classied into three types. An isolated vertex is a contour of length one and an isolated vertex has no in-between edge. In Figure 3, the vertex A is an isolated vertex. Practice shows that layered decomposition rarely produce isolated vertices. We define a vertex as a branching vertex if it is connected to more than two vertices in the same vertex layer. The four vertices A, B, C and D in Figure 4 are all branching vertices. All the left ver-

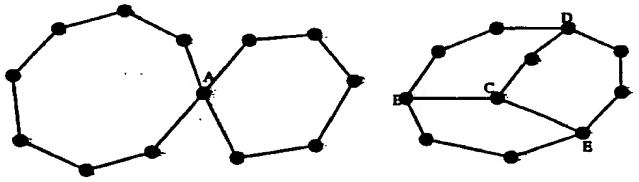


Figure 4: Branching vertices

tices are the ordinary vertices. It is clear that an ordinary vertex usually has two in-between edges. An exception case is that it has only one in-between edge if it is an endpoint of an open contour.

Generalized Triangle Strip We now introduce the basic geometric primitive, the generalized triangle strip, which is at the core of our compression scheme. The generalized triangle strip is used to represent the set of triangles that lie between two adjacent vertex layers. Figure 5 shows a comparison between a triangle strip (the top one) and a generalized triangle strip (the center one). Note that an edge of the mesh either connects two vertices in the same layer (in-between edges) or two vertices in two consecutive layers (chords). Typically, a triangle has one vertex in one vertex layer and the other two vertices in an adjacent vertex layer. Focusing on the strip that connects the vertices of layer i to the vertices of layer $i+1$, we mark with a 0 all the triangles that have two vertices on the i^{th} layer (we will call it parent march) and with a 1 all the triangles that have two vertices in the $(i+1)^{th}$ layer (child march). A regular triangle strip has the sequence of triangle marks that alternate continuously between 0 and 1 so that the sequence of marks does not need to be stored since the strip is completely defined by the position of its starting triangle. In a generalized triangle strip this is no longer true. Once one has the vertex layers, a bit string (one bit per triangle) is needed to encode the generalized triangle strip. We will call this bit string (marks) as a bit march.

Exceptional Triangle Strips and The Rule To achieve the best compression ratio one needs to get the best encoding for the most frequent case. For this reason we do not adopt the triangle strip as a basic primitive since it is not frequent enough. Using a layered decomposition of the vertices of the mesh as described in the next section we obtain a large number of long generalized triangle strips. Hence the generalized triangle strip seems to be a good basic geometric primitive to use for the compact encoding of the mesh. Figure 5 contains two kinds of triangle strips.

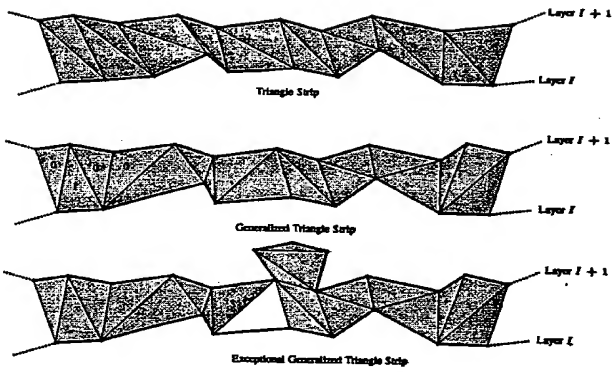
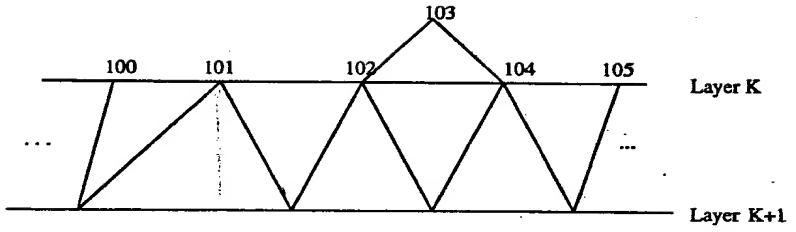


Figure 5: Comparison among a triangle strip (top) a generalized triangle strip (center) and an exceptional generalized triangle strip (bottom).

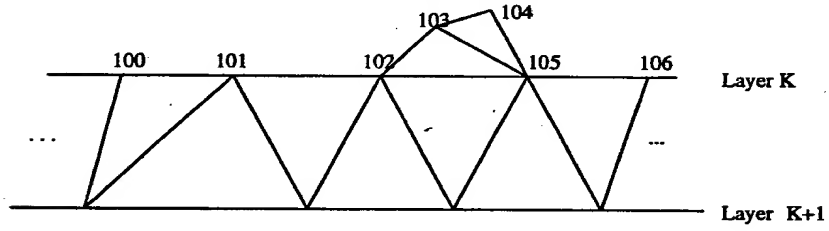
However, the layered decomposition may produce more complicated strips, as shown in Figure 6.

We call the strips in Figure 6 exceptional triangle strips because there are special triangles or holes assocociated with them. In the first case, the triangle $(102, 103, 104)$ is a special triangle whose three vertices are all located in the parent vertex layer (Layer K). In the next case, both $(102, 103, 105)$ and $(104, 105, 102)$ are special triangles. In the last case, $(102, 103, 104)$ is not an existing triangle which we call a hole. In the following, we will call a triangle special if all its three triangles are in the same vertex layer. A bubble will be a set of special triangles associated with a strip.

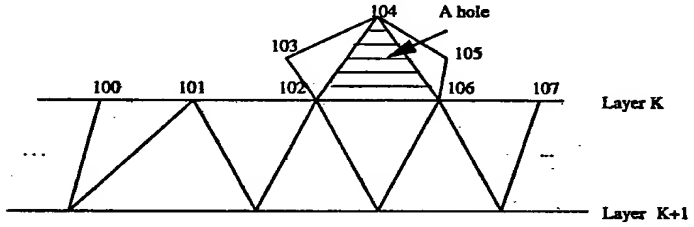
To encode a regular triangle strip (without any associated bubble), the two starting vertices in the parent and child contours and the bit march decide the whole connectivity information. However, it is impossible to express an exception triangle strip with bubbles simply by its starting vertices and bit march because the jumps of bubbles over the parent contour



(a) case 1: a one-triangle bubble



(b) case 2: a two-triangle bubble



(c) case 3: a hole bubble

Figure 6: Exceptional triangle strips

exist.

In order to encode an exception triangle stripe efficiently, we introduce the relative indexing concept. Relative index is defined within a bubble. Suppose that a bubble spans n vertices and the index of the starting vertex of the bubble is X. So the indices of these vertices are $X, X+1, \cdots, X+n-1$. Then their relative index are defined as $0, 1, \cdots, n-1$. All special triangles will be encoded by a triple of relative indices. For instance, the relative indices of 102, 103, 104, 105 in the case 2 of Figure 6 are 0, 1, 2, 3. The two special triangles are encoded as $(0, 1, 3)$ and $(1, 2, 3)$. Since n is usually a small number, encoding special triangles in this way is efficient because many bits are saved. Overall, to encode a triangle strip, the follow values must be included: the two starting vertices of the parent contour and the child contour, the bit march, the number of bubbles, and the bubbles. The two starting vertices will be expressed by two local indices, the bit march will be encoded by Huffman coding (see the connectivity encoding paragraph below), and the bubble encoding.

Global indexing is quite obvious: the vertices in the first vertex layer are numbered first, then vertices of the second vertices are numbered, the numbering process is repeated until all vertices of the last vertex layer are numbered. Suppose that V is the number of vertices of the given mesh. Then $V-1$ is the largest index which is in actual the global index of the last numbered index in the last vertex layer. Global indexing usually requires many bits to encode each vertices because V is large for huge models. Now we introduce the concept of local indexing. For each vertex layer, we will increasingly number all its vertices from 0. First all the branching vertices (if there is any) are numbered. Then for every contour in that layer, we number all non-branching vertices. The numbering order follows the order the contour is formed (counter-clock wise). Lastly, all isolated vertices are numbered. Suppose that L is the number of vertices in a vertex layer, then the local index of any vertex in that

layer will be less than L. Since L is much smaller that V, fewer bits can be used to encode a local index than a global index. Furthermore, the number of bits to encode a local index can be computed easily. It is the smallest positive integer K such that $2^K \geq L$. We want to point out here that there is a one-to-one correspondence between global index and (layer, local index) pair. The global index of a vertex in layer K is equal to the vertex's local index plus the number of vertices of the first $K-1$ vertex layers.

Geometry data (or other attribute data) will be encoded layer by layer, and in each layer by the local vertex numbering order. This is a much natural encoding order. It is an advantage over existing encoding schemes (see [17]) since no vertex spanning tree is encoded.

Figure 7 shows how to encode a bubble associated with a triangle stripe. In the triangle

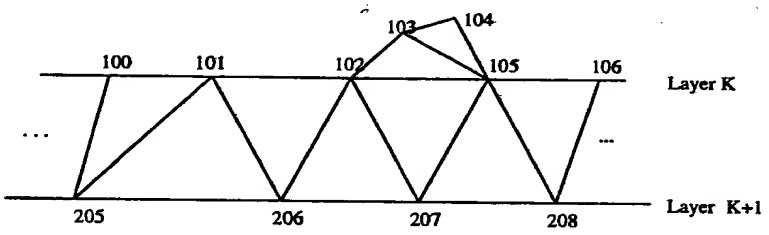


Figure 7: The coding of a exceptional strip

$(102, 207, 105)$ is not a special triangle the bit march still uses "1" to express this special parent march. We call this kind of triangles as jumping triangles. When recovering the connectivity, the decoder must know that there is a bubble when it processes to meet the chord $(102, 207)$. It also needs to how many vertices the bubble involve in the parent contour. So to encode a bubble, the starting edge of the bubble (in this instance, edge $(102, 207)$), the vertex span (the number of involved vertices) of the bubble in the parent contour. As said above, all triangles in the bubble will be encoded by relative indices (not even local indices

). Of course, the encoder needs to inform how many special triangles in the current bubble by a very small number of bits (decided by the vertex span). Just to avoid any possible confusion, the bit march of the exception triangle strip is still $\cdots 1010101 \cdots$.

Therefore, encoding a strip (whether exceptional or not) includes: (1) the two starting vertices by local indices; (2) the bit march (including length); (3) bubbles (including the number of bubbles) by local and relative indices. These information is enough for the decoder to recovery the original triangle strip.

Finally, for each vertex layer, there are another kind of triangles whose vertices are all in the current vertex layer and are in different contours. Figure 8 gives some of these triangles.

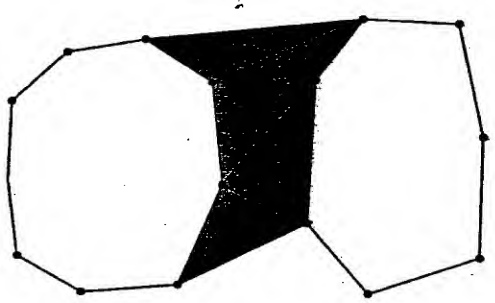


Figure 8: Triangles that span more than one contour in same vertex layer

In Figure 8 all vertices are in the same vertex layer. In our connectivity scheme, each of such special triangles are encoded by a triple of local indices.

No matter what kind of special triangles, their encoding cost is relative large. However, practices show that these triangles only occupy less than one percent of the total triangles. So they will not affect the final compression ratio. For large models, our overall connectivity cost can be as good as 1.1 bits per triangle. This result is obtained by taking a very tricky

encoding scheme described in next paragraph.

Connectivity Encoding Now the connectivity encoding is reduced to encoding all generalized triangle strips, and possibly some special triangles. Basically, there are two kinds of special triangles in the layering mesh structure. The first kind of special triangles are those whose three vertices are located in a same contour (thus same vertex layer). The second kind of special triangles are those whose vertices are in a same layer, but in different contours. To efficiently express special triangles associated with a strip, we use a novel relative indexing technique described in last paragraph . A special triangle in a bubble is encoded by a triple of relative indices. Relative indexing is efficient since it saves space by using less bits for each index in any special triangle. For example, the bunny model has 34,835 vertices. Suppose that there are 500 vertices in current vertex layer. Then, to encode a special triangle in this vertex layer, global indexing requires 48 bits (16 bits per vertex) while local indexing only needs 27 bits (9 bits per vertex). Obviously, too many special triangles are not desirable because they still take up much more space than ordinary triangles (about 1 bit per triangle). Fortunately practices show that special triangles produced by the layering scheme typically occupy only $1-3\%$ of the whole faces.

Now the left problem is how to encode ordinary strips which are expressed through bit marches, strings of 0s and 1s. A simple and direct way to encode a strip is to first encode the length of the strip and then each marching bit by a single bit, either 0 or 1. Although such encoding scheme is simple and easy to implement, it is not efficient. We have found that the patterns 0101 and 1010 appear more frequently in a typical bit march than other patterns such as 1111 or 0000, $\cdots$ if we group every four consecutive bits into a symbol. Therefore there are totally sixteen such symbols, as shown in Table 1, to encode the bit march. The

idea is to group each four consecutive bits in a bit march into a symbol. In case the length of a bit march is not a number of multiple four, the left bits (at most three) can be easily dealt with.

symbol	code	symbol	code
0000	0101011111	1000	01010100
0001	010101110	1001	0100
0010	01011	1010	00
0011	1111	1011	0111
0100	010100	1100	1110
0101	10	1101	0110
0110	110	1110	01010101
0111	01010110	1111	0101011110

Table 1: Static Huffman table for connectivity encoding

Based on many large models, we have made statistics about the average occurrence frequencies of the sixteen symbols and created the Table 1 by the classic Huffman coding algorithm. As shown in this table, symbols 0101 and 1010 appear more frequently, so these symbols use less bits to encode. This connectivity encoding scheme saves about 25% less space.

Table 2 gives results of our connectivity coding cost. Figure 9 shows the results of more than 300 models.

model	nV	nT	CC	CBPV	CBPT
bunny	34835	69473	105216	3.02	1.51
fandisk	6475	12946	19112	2.95	1.47
head	3690	7376	20760	5.62	2.81
eight	766	1536	3016	3.94	1.96
horse	11135	22258	36000	3.10	1.55
shape	2562	5120	4176	1.63	0.82
mannequin	689	1355	4440	6.44	3.27
mechpart	1932	3872	8528	4.41	2.20
patella	780	769	1384	1.77	1.79
phone	83044	165963	169352	2.03	1.02
apache	13025	17673	34992	2.68	1.97
apple	867	1704	2776	3.20	1.62
banana	272	512	1000	3.67	1.95
blades	732	1292	1320	1.80	1.02
canstick	2101	4150	4568	2.17	1.10
chain	840	1680	2872	3.41	1.70
helix	6480	6400	7240	1.11	1.13
human5	4738	8364	13896	2.93	1.66

Table 2: Results for coonectivity coding nV : number of vertiçes; nT : number of triangle faces; CC : cost of connectivity coding (bits); CBPV = CC/nV; CBPT = CC/nT

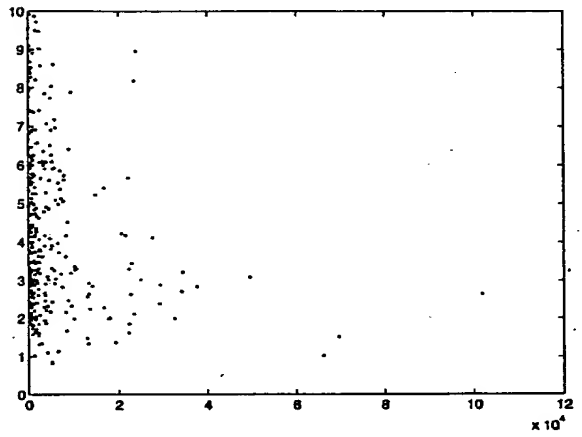


Figure 9: Connectivity coding (horizontal : number of triangles, vertical : cost in bits)

2.1 Geometry Encoding using Vector Quantization

A vector quantizer is a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over constrained bandwidth channels [6], [5], [7]. Vector quantization is a data compression technique to reduce the bit rate so as to minimize communication channel capacity or digital storage requirements with little compromise in the image quality so that necessary fidelity of the data is maintained. In this subsection, a productive vector quantization scheme is introduced in order to efficiently encode vertex coordinates of a given meshed object. Details are provided for the codebook design, nearest neighbor searching, and error/distortion control.

Codebook Design Without loss of generality, suppose that the sample space is

$$\mathbf{S} = \{\mathbf{x} \mid ||\mathbf{x}||_2 \leq 1, \mathbf{x} \in I\!R^3\}$$

Let the order set $C = \{y_1, y_2, \cdots, y_N\}$ be the codebook we are going to design. N is the codebook size which is adjustable according to the user-specified error tolerance. $Y_i, i = 1, 2, \cdots, N$ are the code vectors.

A vector quantizer Q of dimension three and of size N is a mapping from a vector in S to a code vector in C. Basically, every $N-$point vector quantizer has an associated partition of S into N regions or cells, S_i $(i = 1, 2, \cdots, N)$, where the ith cell is defined by

$$S_i = \{x \in S : Q(x) = y_i\}$$

The *code rate* of the vector quantizer Q is $r = (log_2 N)/3$. The code rate measures the number of bits per vector component used to express the input vector. Generally, the code rate shows the accuracy of a vector quantizer, which depends on the goodness of the chosen codebook. Clearly, large-size codebook gives good accuracy but results in a large bit rate which means poor compression performance.

We present here a productive vector quantizer defined by of three sub-codebooks. A vector (x, y, z) in S can be expressed in the sphere coordinates (r, ϕ, θ) by the relation

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2},\ 0 \le r \le 1 \\ \phi &= atan(y/x),\ 0 \le \phi \le 2\pi \\ \theta &= atan(\sqrt{x^2 + y^2}/z)\ 0 \le \theta \le \pi \end{aligned}$$

The three scalar r, ϕ and θ are encoded separately by using the following three scalar sub-codebooks C_r, C_ϕ and C_θ which are of size $N_r = 2^{K_r}$, $N_\phi = 2^{K_\phi}$ and $N_\theta = 2^{K_\theta}$ respectively

$$\begin{aligned} C_r &= \{r_i \mid r_i = i - 1/N_r,\ i = 1, 2, \cdots, N_r\} \\ C_\phi &= \{\phi_j \mid \phi_j = 2\pi(j-1)/N_\phi,\ j = 1, 2, \cdots, N_\phi\} \\ C_\theta &= \{\theta_k \mid \theta_k = \pi(k-1)/N_\theta,\ k = 1, 2, \cdots, N_\theta\} \end{aligned}$$

A vertex thus will be encoded by $K_r + K_\phi + K_\theta$ bits. And the overall codebook can be regarded as

$$C = \{\mathbf{p}_{ijk} := (x_{ijk}, y_{ijk}, z_{ijk})^T \mid i = 1, 2, \cdots, N_r; j = 1, 2, \cdots, N_\phi; k = 1, 2, \cdots, N_\theta\}$$

with $x_{ijk} = r_i \sin\theta_k \cos\phi_j$, $y_{ijk} = r_i \sin\theta_k \sin\phi_j$, $z_{ijk} = r_i \cos\theta_k$.

Nearest Neighbor Searching The squared error will be used as the error measure

$$d(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 = \sum_{i=1}^{3} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

For a given vector $\mathbf{x} \in S$, the task of nearest neighbor searching is to search for its closest code vector in the sense that it has minimal distortion $d(\mathbf{x}, \mathbf{y}_j)$ over all the code vectors in the codebook C. Naive linear searching is of $O(N)$ time complexity. $O(\log N)$ could be achieved by well-organized codebook [6]. For our codebook given above, only constant time is required for a one time nearest neighbor search. For any vector $\mathbf{x} \in S$, its sphere coordinate (r, ϕ, θ) can first be computed in constant time. Then three intervals $[r_i, r_{i+1}]$, $[\phi_j, \phi_{j+1}]$ and $[\theta_k, \theta_{k+1}]$ are found, within which r, ϕ and θ located respectively. Distortions between $\mathbf{x}$ and each of the eight code vectors $\{\mathbf{p}_{i'j'k'}, i' = i, i+1; j' = j, j+1; k' = k, k+1\}$ are computed. Finally, since we know that the distortion must equal one of eight distortion, the code vector can be easily decided by picking out the one which has the minimal distortion. All this can be done in constant time. Let $PVQ(\mathbf{x})$ be that code vector and

$$D_C(\mathbf{x}) = d(\mathbf{x}, PVQ(\mathbf{x}))$$

Error Control Given a user specified error tolerance ϵ, one must decide the sizes, N_r, N_ϕ and N_θ, of the three sub-codebooks. These three parameters should be chosen as small as

possible so as to attain a small bit rate. The following quantitative analysis shows how the error control is directed.

For any vector $\mathbf{x} \in S$, its associated sphere coordinates (r, ϕ, θ) must be located in some cube $[r_i, r_{i+1}) \times [\phi_j, \phi_{j+1}) \times [\theta_k, \theta_{k+1})$. So the distortion will not exceed half the length of the diagonal of that cube. $\mathbf{p_0} = (r_i, \phi_j, \theta_k)$ and $\mathbf{p_1} = (r_{i+1}, \phi_{j+1}, \theta_{k+1})$ are two endpoints of a diagonal. Suppose that $(\mathbf{x_0}, \mathbf{y_0}, \mathbf{z_0})$ and $(\mathbf{x_1}, \mathbf{y_1}, \mathbf{z_1})$ are their Cartesian coordinates. Since

$$\begin{aligned}
|\mathbf{x_1} - \mathbf{x_0}| &= (r_i + \tfrac{1}{N_r}) \sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos(\phi_j + \tfrac{2\pi}{N_\phi}) - r_i \sin\theta_k \cos\phi_j \\
&= r_i(\sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos(\phi_j + \tfrac{2\pi}{N_\phi}) - \sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos\phi_j + \sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos\phi_j \\
&- r_i \sin\theta_k \cos\phi_j) + \tfrac{1}{N_r} \sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos(\phi_j + \tfrac{2\pi}{N_\phi}) \\
&= r_i(\sin(\theta_k + \tfrac{\pi}{N_\theta})(-2\sin(\phi_j + \tfrac{\pi}{N_\phi}) \sin(\tfrac{\pi}{N_\phi})) \\
&+ 2\cos(\theta_k + \tfrac{\pi}{2N_\theta}) \sin(\tfrac{\pi}{2N_\theta}) + \tfrac{1}{N_r} \sin(\theta_k + \tfrac{\pi}{N_\theta}) \cos(\phi_j + \tfrac{2\pi}{N_\phi})
\end{aligned}$$

we have

$$|\mathbf{x_1} - \mathbf{x_0}| \leq r_i(\frac{2\pi}{N_\phi} + \frac{\pi}{N_\theta}) + \frac{1}{N_r} \leq \frac{2\pi}{N_\phi} + \frac{\pi}{N_\theta} + \frac{1}{N_r}$$

Similarly,

$$|\mathbf{y_1} - \mathbf{y_0}| \leq \frac{2\pi}{N_\phi} + \frac{\pi}{N_\theta} + \frac{1}{N_r}, \ |\mathbf{z_1} - \mathbf{z_0}| \leq \frac{\pi}{N_\theta} + \frac{1}{N_r}$$

Theorem 2.1 *If N_r, N_ϕ and N_θ satisfy*

$$\frac{1}{N_r} + \frac{2\pi}{N_\phi} + \frac{\pi}{N_\theta} < \frac{2}{\sqrt{3}}\epsilon$$

then for any $\mathbf{x} \in S$ $D_C(x) < \epsilon$.

Geometric Encoding without Accumulating Error The layer decomposition technique described above groups all vertices by layers. Each layer consists of one or more contours which could be either closed or open. So the problem of geometry encoding is

now reduced to the problem of encoding the geometry of vertices in a single contour. In this paragraph, we will describe an efficient method that can encode coordinates without accumulating error, the distortion from productive vector quantization.

Let $P_0, P_1, \cdots, P_n$ be the vertices which form a contour in that order. Define $\Delta P_i = P_{i+1} - P_i$ for $i = 0, \cdots, n-1$. Using linear prediction, a straight way to encode all the vertex coordinates would be (1) encode P_0 directly; (2) encode $\Delta P_i, i = 0, \cdots, n-1$ by the productive vector quantization method. But there are two disadvantages which are associated with this scheme. On one hand, the error from the productive vector quantization can be accumulated. On the other hand, redundance still exists in this raw encoding scheme. The accumulating error problem can be solved by taking an on-the-fly computing skill so that any vertex after recovered by the decoder will not deviate its original position by a Euclidean distance more than the maximum distortion of the vector quantization. To overcome the second disadvantage, a very tricky yet intuitive scheme is designed. This scheme is based on statistics of a general case behavior of correction vectors ΔP_i in ordinary contours of our layer decomposition structure.

In order to illustrate the first problem, let ϵ_i be the distortion of ΔP_i from the productive vector quantization. Let $\Delta \bar{P}_i = \Delta P_i + \epsilon_i$. $\Delta \bar{P}_i$ is the actual code vector. The decoder only knows this code vector $\Delta \bar{P}_i$. It does not know neither ϵ_i nor ΔP_i. Let $\bar{P}_k$ be the corresponding position of P_k that is recovered. Then

$$\begin{aligned}
\bar{P}_{i+1} &= \bar{P}_i + \Delta \bar{P}_i \\
&= \bar{P}_{i-1} + \Delta \bar{P}_{i-1} + \Delta \bar{P}_i \\
&= \bar{P}_0 + \textstyle\sum_{k=0}^{i} \Delta \bar{P}_k \\
&= \bar{P}_0 + \textstyle\sum_{k=0}^{i} (\Delta P_k + \epsilon_k) \\
&= P_{i+1} + \textstyle\sum_{k=0}^{i} \epsilon_k
\end{aligned}$$

Therefore, the accumulating error for i^{th} vertex in a contour is $\sum_{k=0}^{i} \epsilon_k$. This kind of accumulating error destroys the position fidelity of vertices with large i. To solve this problem, we design a scheme which guarantees that the deviation of any vertex position is controlled within the distortion of the productive vector quantization, with the reasonable assumption that $\bar{P}_0 = P_0$ because one can always fully encode the first single vertex with more bits (say, 16 bits) without significantly affecting the final compression ratio. The main idea of the following refined codes is that once a correction vector is quantized, its distortion will be merged into its following correction vector. That is to say, we encode $\Delta P_i + \epsilon_{i-1}$ instead of ΔP_i. The whole refined scheme is

1. Encode P_0 directly
2. (a) Encode ΔP_0 directly; compute $\hat{\epsilon}_0 = \Delta\bar{P}_0 - \Delta P_0$ with $\Delta\bar{P}_0$ being the code vector of ΔP_0
   (b) For $i = 1, \cdots, n-1$, encode $\Delta P_i + \epsilon_{i-1}$; compute $\epsilon_i = \Delta\bar{P}_i - (\Delta P_i + \epsilon_{i-1})$ with $\Delta\bar{P}_i$ being the code vector of ΔP_i.

Claim 2.1 $|\bar{P}_i - P_i| \leq \epsilon$, *for* $i = 0, 1, \cdots, n$. *Here* ϵ *is the maximum distortion possible.*

This can be justified quite straightforward. First of all, P_0 is encoded directly so the introduced error can be neglected. For $i = 1$, since ΔP_0 is encoded directly, we have $|\Delta\bar{P}_0 - \Delta P_0| = |\epsilon_0| \leq \epsilon$. By definition, $\bar{P}_1 = \bar{P}_0 + \Delta\bar{P}_0$ and $P_1 = P_0 + \Delta P_0$, then

$$\begin{aligned} |\bar{P}_1 - P_1| &= |\bar{P}_0 - P_0 + \Delta\bar{P}_0 - \Delta P_0| \\ &\leq |\bar{P}_0 - P_0| + |\Delta\bar{P}_0 - \Delta P_0| \\ &\leq \epsilon \end{aligned}$$

Similarly, for $i \leq 1$, since $\Delta\bar{P}_k = (\Delta P_k + \epsilon_k)$, we have

$$\begin{aligned}
\bar{P}_{i+1} &= \bar{P}_i + \Delta\bar{P}_i \\
&= \cdots \\
&= \bar{P}_0 + \textstyle\sum_{k=0}^{i}(\Delta P_k + \epsilon_{k-1} - \epsilon_k) \ \ (\epsilon_{-1} := 0) \\
&= \bar{P}_0 - P_0 + P_{i+1} - \epsilon_i \\
&= P_{i+1} - \epsilon_i
\end{aligned}$$

Therefore

$$|\bar{P}_{i+1} - P_{i+1}| \leq |\epsilon_i| \leq \epsilon$$

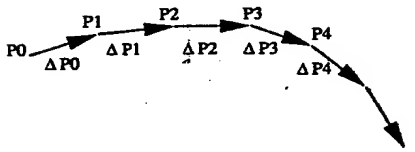


Figure 10: Redundance within adjacent correction vectors

Figure 10 shows the redundance contained within adjacent correction vectors. Suppose ΔP_k and ΔP_{k+1} are two adjacent correction vectors and the two integer triples (r_k, ϕ_k, θ_k) and $(r_{k+1}, \phi_{k+1}, \theta_{k+1})$ are their corresponding codes. We observe that since the angular change in 3D from ΔP_k to ΔP_{k+1} is not much in general, so $\phi_{k+1} - \phi_k$ and $\theta_{k+1} - \theta_k$ should usually be small integers. Table 3 manifests this fact. In this table, N is not fixed which is 32 for Δr and $\Delta\theta$, and 64 for ϕ. It must be pointed out that the true correction values Δr, $\Delta\phi$ and $\Delta\phi$ could be negative. Is there any need to spend one extra bit to indicate the sign of each correction value? The answer is no. We simply add an N if the correction value is negative. The decoder must take care of this convention by making sure that the recovered values r, ϕ and θ be within the range $[0, N-1]$. Here r, ϕ and θ mean the integer codes, not

the true float values. An simple example may help to understand this trick. Suppose that we want to encode r_1 and r_2, the integer codes of the lengths of two consecutive correction vectors. Also assume $r_1 > r_2$. Thus $\Delta r = r_2 - r_1 < 0$. Now we encode $\Delta r + N$ which is in the interval $[0, N-1]$, instead of encoding Δr which is negative. After having recovered r_1 and the correction value $\Delta\bar{r}$, the decoder first computes $\bar{r}_2 (= r_1 + \Delta\bar{r})$. If $\bar{r}_2$ is in the interval $[0, N-1]$, then r_2 equals to $\bar{r}_2$. In our case, $\bar{r}_2 >= N$, which means that $\Delta\bar{r}$ is the summation of the true difference plus N. So r_2 equals to $\bar{r}_2 - N$ according to the fact that $\bar{r}_2 = r_1 + (r2 - r_1 + N) = r_2 + N$. This trick can be similarly applied to the other two components ϕ and θ.

object		symbols								
		0	1	2	3	N-3	N-2	N-1	others	total
fandisk	Δr	1662	1074	247	166	161	297	1111	1585	6303
	$\Delta\phi$	1840	794	393	139	118	376	798	1845	6303
	$\Delta\theta$	3009	710	164	92	71	101	541	1615	6303
bunny	Δr	8108	5325	1661	1600	1659	1627	5473	8729	34182
	$\Delta\phi$	7464	3817	1917	762	822	2044	3917	13439	34182
	$\Delta\theta$	7715	5081	2320	1058	1110	2053	4556	10289	34182
phone	Δr	21259	13759	3715	3037	3003	3624	13803	19926	82126
	$\Delta\phi$	14811	11796	4462	1900	2404	5091	12470	29192	82126
	$\Delta\theta$	20827	12013	4192	2094	2100	4037	12017	24846	82126

Table 3: Frequency statistics of the code symbols

Finally, we have the following encoding scheme for a single contour.

1. Encode P_0 directly

2. (a)
   - Encode ΔP_0 directly by its PVQ code triple (r_0, ϕ_0, θ_0);
   - Compute $\epsilon_0 = \Delta \bar{P}_0 - \Delta P_0$ with $\Delta \bar{P}_0$ being the code vector of ΔP_0

   (b) For $i = 1, \cdots, n-1$,
   - Compute the PVQ code triple (r_i, ϕ_i, θ_i) of the vector $\Delta P_i + \epsilon_{i-1}$;
   - Encode $(r_i - r_{i-1} + N_r) \bmod N_r$, $(\phi_i - \phi_{i-1} + N_\phi) \bmod N_\phi$, $(\theta_i - \theta_{i-1} + N_\theta) \bmod N_\theta$;
   - Compute $\epsilon_i = \Delta \bar{P}_i - (\Delta P_i + \epsilon_{i-1})$ with $\Delta \bar{P}_i$ being the code vector of ΔP_i.

Note that $\Delta \bar{P}_i$ can be obtained from the three codebooks.

In our implementation, we designed a modified version of the classic Huffman coding [9]. The three components Δr, $\Delta \phi$ and $\Delta \theta$ are encoded separately with three different Huffman tables. Let S be the symbol set of all integers in $[0, N-1]$. Again N varies, dependent on the component being encoded. Notice here that every symbol is an integer. If the whole encoding process is off-line, then the exact occurrence of each symbol can be computed. Therefore, the classic Huffman codes can be utilized. However, classic Huffman codes has two obvious disadvantages. First, the statistics procedure is time-consuming for very large input models and thus is not suitable for many applications. The second disadvantage is that the encoder needs to transfer a large Huffman table which will surely affect the final compression ratio. Our algorithm is based on a static Huffman table which is created by measuring symbol occurrence from many large models. This algorithm is very efficient because there is no need to send the Huffman table. Furthermore, the speed is fast. Dynamic Huffman coding can also be used.

Results show that code difference encoding is more efficient than direct encoding the PVQ codes of the prediction vectors. If we regard each integer code as a symbol, Figures

11, 12 and 13 show the frequencies of symbols for three different size of models. In this figure, each solid curve stands for the frequency of code differences while the dash line is for codes from direct encoding scheme.

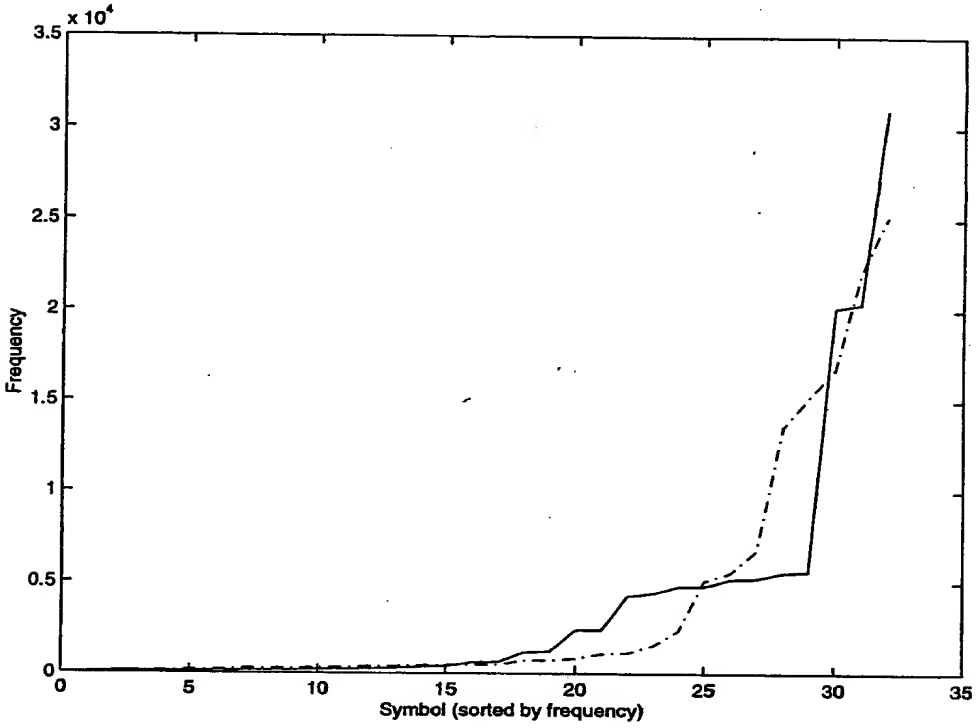


Figure 11: frequency : r

From these frequency curves, it is not hard to find that in the code difference approach most codes are accumulated around only a few symbols, mostly at the two boundary area of the code spaces which are $[0, 31]$, $[0, 63]$ and $[0, 32]$ in our example figure. This observation is also supported by Figure 10. Recall the fact that the code differences between codes of

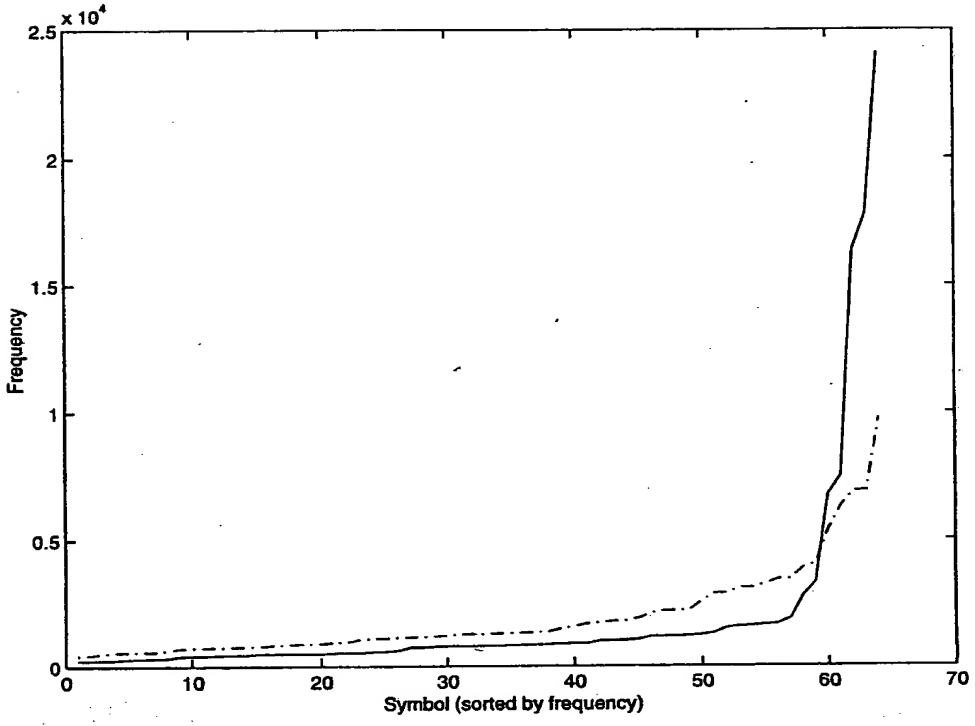


Figure 12: frequency : ϕ

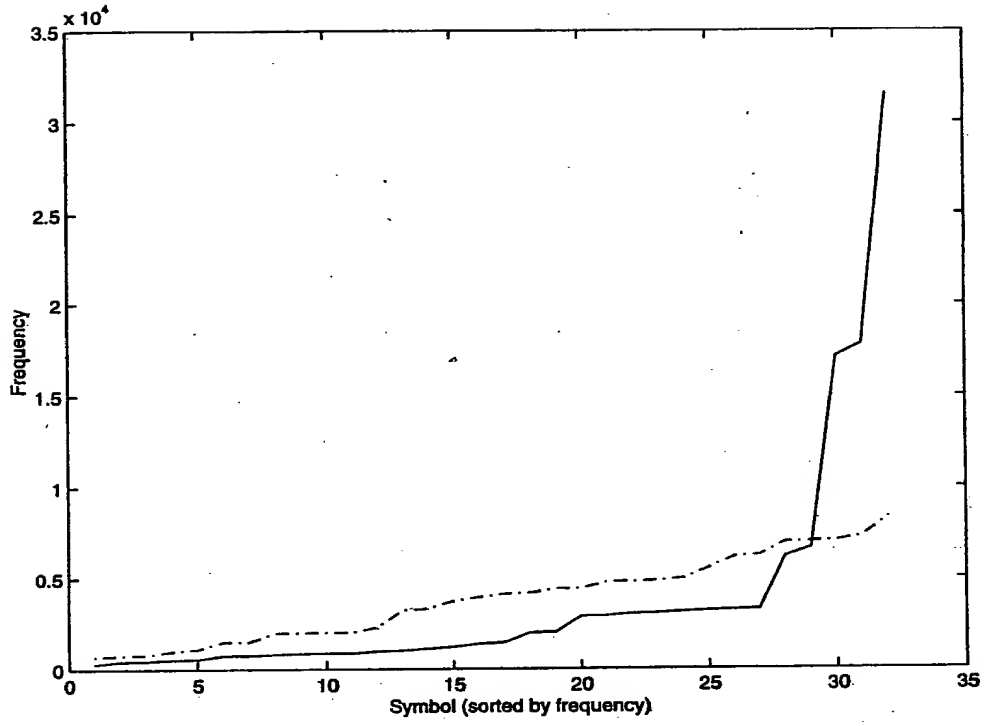


Figure 13: frequency : θ

adjacent correction vectors are usually small. Table 4 compares entropy for code difference and that for direct coding with $N_r = 32$, $N_\phi = 64$ and $N_\theta = 32$. Definition of entropy is as follows. Suppose that a message consists of the symbols set $\{S_i\}_{i=0}^{n}$, with p_i being the occurrence frequency (probability) of symbol S_i. Then entropy of the message is defined as

$$\rho = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$$

For entropy coding such Huffman and arithmetic coding, the smaller the entropy, the fewer the number of bits needed to a symbol averagely. In Table 4, direct coding is actual Deering original work on geometry encoding which is considered to be a state-of-the-art method. Our scheme improves it in this aspect by about 15% storage space even by the static Huffman coding.

Figure 14 - 18 are graphs in which objective evaluation is given based on simulation of over 300 models. The target bit rates are 10, 15, 20, 25 and 25 in term of bits per vertex. The actual bit rate for each model must be record within 1% margin.

In summary, the productive vector quantizer encodes vertex coordinates with the following advantages. First, the codebook is small and thus does not occupy much memory. In addition, only the size of the codebook is needed at the decoding side and the entire codebook does not need to be transmitted. Secondly, the constant time nearest neighbor searching makes the encoder (and also the decoder) perform extremely fast. This feature cannot be reached by an arbitrary length vector quantization. Finally, this vector quantization method can attain any specified error tolerance by adjusting the sizes of the sub-codebooks. Our encoding scheme guarantees no accumulating errors by the skill of merging current distortion in coming correction vector. Furthermore, by encoding code difference, our scheme improves the state-of-the-art geometry encode algorithm originated in [3].

		Code Difference (bits)	Direct Coding (bits)
Fandisk	R	3.81	4.58
	P	4.23	4.99
	T	3.28	4.02
	total	11.32	13.59
Bunny	R	3.68	3.64
	P	4.64	5.44
	T	4.05	4.81
	total	12.37	13.89
Phone	R	3.48	3.58
	P	4.61	5.57
	T	3.89	4.72
	total	11.99	13.87

Table 4: Entropy comparison

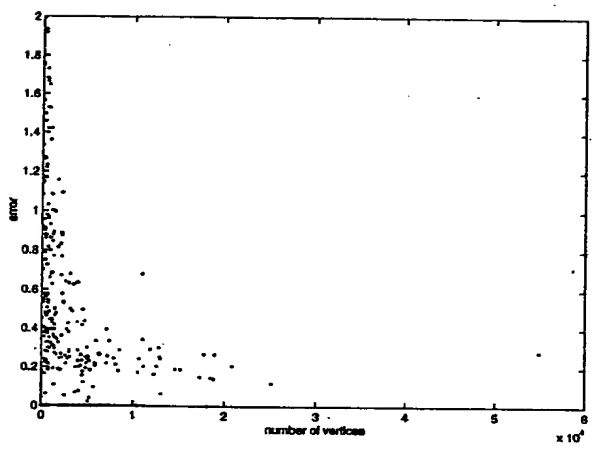


Figure 14: target bit rate: 10

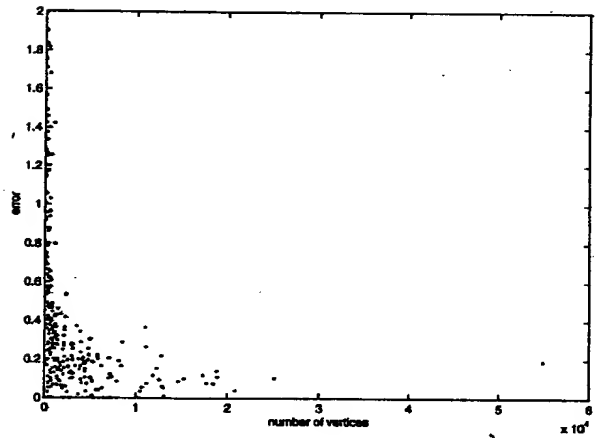


Figure 15: target bit rate: 15

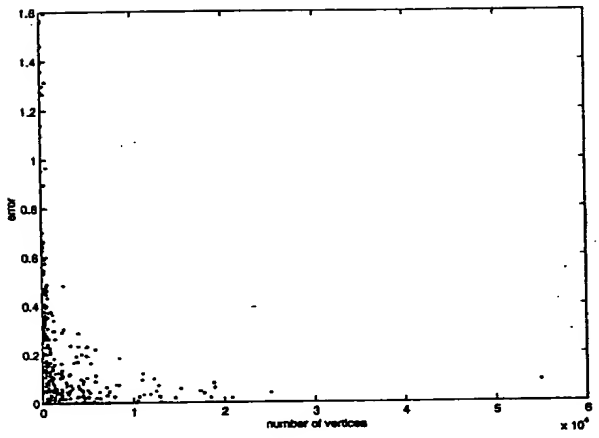


Figure 16: target bit rate: 20

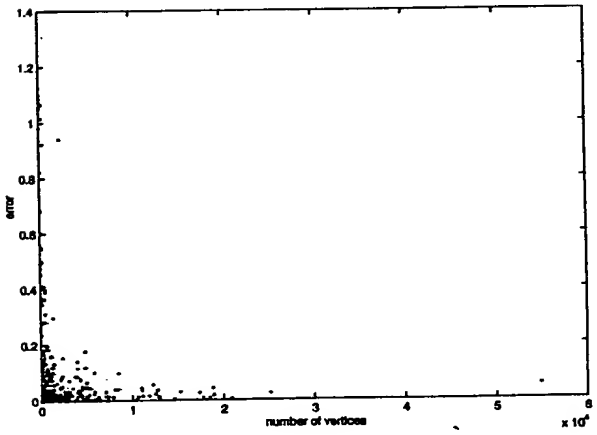


Figure 17: target bit rate: 25

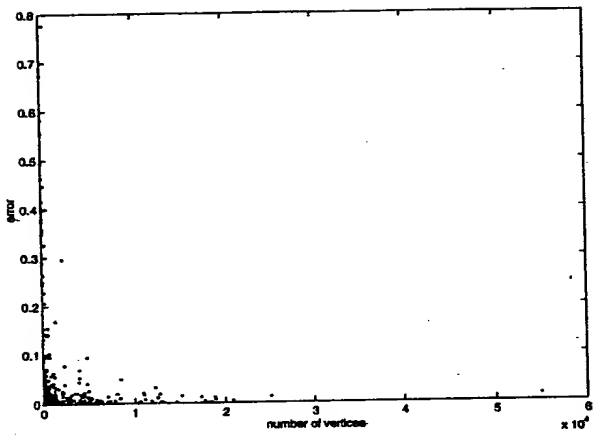


Figure 18: target bit rate: 30

3 Progressive Transmission

In this section, we will describe progressive transmission in two directions: progressive bit transmission and progressive connectivity transmission.

3.1 Progressive Bit Transmission

Without loss of generality, assume that only the vertex coordinate data is to be encoded and the given mesh is connected. In order to apply our productive vector quantization, we do a normalization step over coordinates of all vertices. After normalization, the Euclidean length of each coordinate is smaller than 0.5. We require that the normalization factor should be a integer of 2^K. The integer K is encoded.

Recall that vertices of the original mesh are grouped layer by layer; within any layer, vertices are further grouped by contours; and the first layer consists of a single vertex. Geo-

metric encoding is then quite straight forward. Fully encode all the three coordinate components of the only vertex in the first layer, say, each with 32 bits. For the other vertices, linear prediction is used. The encoding order is important. Here we first give the relationship between vertices. The vertex in the first layer is taken as the root. For each contour in the second layer, choose one proper vertex and regard it as one child of the root; in some fixed direction (for example, counter-clock-wise) the other vertices are encoded successively; for any two adjacent vertices within a contour, the first encoded vertex is regarded as the parent of the second one. For the other layers, a parent-child relationship can be built similarly. Overall, a vertex may have many children but has one and only one parent (the root is an exception). This relationship can be encoded quite efficiently because for any contour only the parent of the starting vertex (the first encoded vertex) must be encoded by a pair of integers: one indicates which contour the parent is located; the other shows the index of the parent within that contour.

With linear prediction, the correction for each vertex is defined as the coordinate difference between this vertex and its parent. One can expect the correction is a vector with three small components since edges in the original mesh are usually short. At least, the length of this correction is less than one, considering our normalization step. Productive vector quantization can be used for each correction. The resulting code vector is expressed by three indices pointing to the three sub-codebooks respectively. Indices can be encoded either directly or by further using entropy coding (Huffman codes [9], Arithmetic coding [19]).

Geometric encoding in the above way must consider error accumulations. In order to avoid this, an on-the-fly computing scheme has been adopted. The principle is simple: to compute the correction the parent coordinates are taken as the one that will be taken while decoding, not the exact parent coordinates.

Progressive transmission of 3D compressed objects can be executed in two main directions. Progressive mesh [14] provides a way of connectivity refinement. A coarse level object can be updated once enough details have come. The other direction is to perform successive approximation to the geometry data (we consider vertex positions). Each coordinate may be encoded in several stages, according to the bit significance status of a float point or its residue. New residue is computed if some bits have been encoded and sent out. To encode a whole mesh, coordinates are first successively quantized. At the first stage, most significant bits are clustered together and each data is encoded by some fixed number of bits. The second stage is performed similarly, but this time on the most significant bits in the residue data. Once all bits in the original data have been sent out, the progressive procedure finishes.

In such a progressive transmission method using successive approximation, all connectivity data is expected to be encoded at the first stage and will not be updated any more. This is not a problem at all since connectivity encoded (symbolic) data usually occupies a small portion of the overall compressed data size in comparison to geometric encoded (numeric) data. In all our experiments the connectivity encoded data was always less than 20% of the overall compressed encoded data size (greater than 80% is for geometry).

3.2 Connectivity Progressiveness

The progressive connectivity transmission fully depends on the connectivity decomposition of meshes. Two kinds of connectivity decomposition will be included: intra-layer decomposition and inter-layer decomposition. Roughly speaking, intra-layer decomposition eliminates about half of the vertices within each layer. Inter-layer decomposition eliminates half of the vertex layers in the mesh. In order to avoid appearing very slim triangles in the

simplified model, intra-layer decomposition and inter-layer decomposition are performed alternatively.

Intra-Layer Connectivity Decomposition Our strategy of intra-layer connectivity decomposition is as follows. In one intra-layer decomposition procedure, half vertices of each vertex layer will be decimated. Vertices in each contour are decimated alternatively. Suppose that all vertices in a contour are numbered consecutively, then we can put all vertices with odd numbers as the vertices to be eliminated. As a result of vertex reduction, the connectivity of the mesh changes. For each triangle strip, its bit march will be decomposed into a coarse level bit march and a couple of details. This decomposition is further divided into two phases. The first phase eliminates the candidate vertices in the child contour. Vertices in each contour are decimated by some fixed order, either counter-clock wise or clock wise. Vertex elimination should be understood as vertex merging: any decimated vertex is regarded as being merged to its front vertex. All incident edges in that strip move with the merged vertex. That is, these edges will have its front vertex as one end point. The detail associated with a removed vertex is the number of its incident edges. For efficiency, we will still express the details by bit strings. If the incident edge number is m, then we use a string of length m to express it. The string is $m-1$ '1' bits followed by a '0' bit. The '0' bits is used as a separator. Here we have made use of the fact that $m \geq 1$. The second phase, performed on the parent contour, is just the same as above. An example below should make it more clear.

Figure 19 gives an example of a simple strip whose bit march string is "0110100010 $\cdots$". Intra-layer decomposition will eliminate those vertices which are marked with "X" and obtain a coarse level triangle strip.

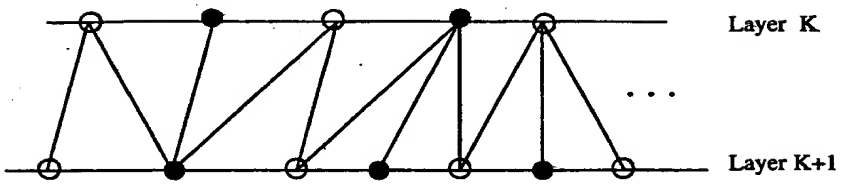


Figure 19: A simple triangle strip

The first step is to eliminate the black vertices in the child contour (Layer K+1). The second step is to eliminate the black vertices in the parent contour (Layer K). Elimination is performed by merging a black vertex into its front white vertex. At the same time all its incident edges will moves with it. The detail related to this merged vertex will record this edge movement.

Figure 20 shows the strip after the first phase of the intra-layer decomposition Now the coarse level bit march is "1101010$\cdots$". And the detail is "11000$\cdots$". "110" is the detail part that is associated with the first removed vertex. Since there are totally three edges move with it, we use "11" to express. "0" is used to express the end of the detail part. Since there is only one edge moving with the second and the third removed vertex, the detail parts associated them are both "0".

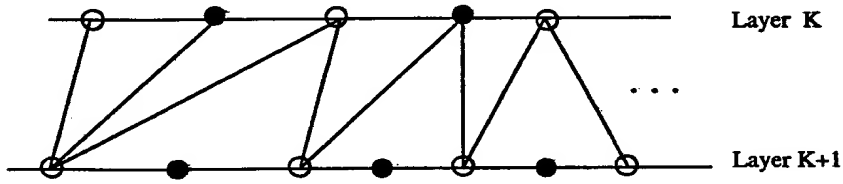


Figure 20: The simplified triangle strip after the first step of intra-layer decomposition

Based on the simplified strip in Figure 20, the second phase of the intra-layer decompo-

sition is performed. The final coarse level strip is shown in Figure 21. And its bit march is "10010$\cdots$" and the detail is "010$\cdots$"

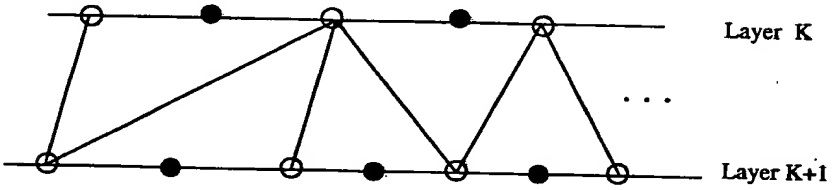


Figure 21: The simplified triangle strip after step two of intra-layer decomposition

Let

$$C = 0110100010\cdots$$
$$C_1 = 1101010\cdots$$
$$D_1 = 11000\cdots$$
$$C_2 = 10010\cdots$$
$$D_2 = 010\cdots$$

where C is the original bit march, C_1 and C_2 are bit marches of the two coarse level strips, D_1 and D_2 are the details. We have

$$C = C_1 \oplus D_1$$
$$C_1 = C_2 \oplus D_2$$

with the operator $\oplus$ being understood as the corresponding reconstruction operation which is quite straightforward. Therefore,

$$C = C_2 \bigoplus D_2 \bigoplus D_1$$

If k consecutive steps of intra-layer decomposition are performed, we have

$$C = C_{2k} \bigoplus D_{2k} \bigoplus D_{2k-1} \bigoplus \cdots \bigoplus D_2 \bigoplus D_1$$

The above formula looks like the function space decomposition defined by wavelets. In fact, the intra-layer decomposition has built a multiresolution mesh based on the layering structure.

Intra-Layer Connectivity Reconstruction The intra-layer connectivity reconstruction procedure is pretty clear. When a vertex is added back to a contour, the local connectivity changes as in Figure 22. Note that its detail is used to move the incident edges back to the new added vertex.

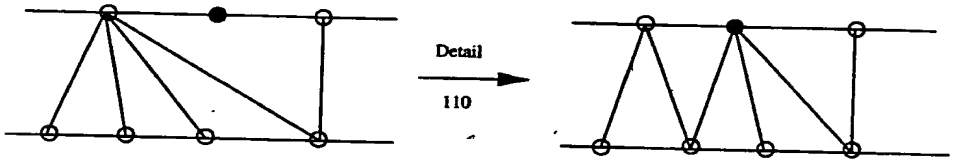


Figure 22: Intra-layer connectivity reconstruction

Inter-Layer Connectivity Decomposition If only intra-layer decomposition is performed, most triangles will become too slim (bad aspect ratio). So there is a need for inter-layer decomposition.

Inter-layer decomposition will eliminate a whole vertex layer. Figure 23 shows two triangle strips with a common contour at Layer K. The bit march of the upper strip is "$010101001100\cdots$". The bit march of the lower strip is "$10100101010011\cdots$". Layer K is the candidate to be eliminated.

The simplified strip will have layer $K-1$ as its parent contour and $K+1$ as its child contour. The problem is how to express the details. Our strategy is as follows. The vertices in layer K are eliminated one by one. When one vertex is to be eliminated, its boundary

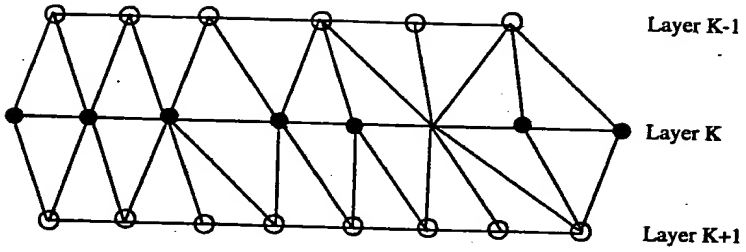


Figure 23: Two adjacent triangle strips

is first decided. The boundary is a 3D polygon. Its related detail part is vertex positions (connectivity positions, not geometry) in the parent contour and the child contour. The positions of the leftmost vertex and rightmost vertex in the parent contour are needed. So are the positions of the leftmost vertex and rightmost vertex in the child contour. For the sake of simplicity, we introduce the concept of sweep edge. A sweep edge of a candidate vertex is the line segment which connects two leftmost vertices in its boundary, one in the parent contour, one in the child contour. In Figure 24, edge $e1$ is the sweep edge of vertex 1, and dashed $e2$ the sweep edge of vertex 2.

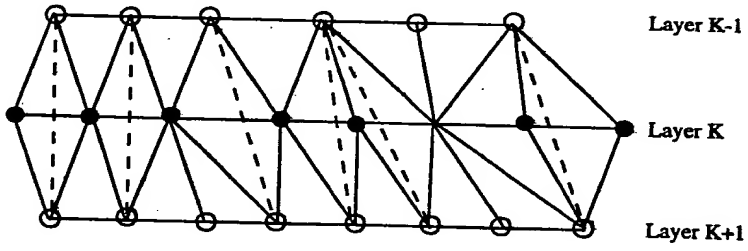


Figure 24: Sweep edges

Once a vertex is removed, there leaves a hole. Local triangulation is then performed. For a eliminated vertex, its detail part must record how the sweep line changes. So it is enough to

use the number of coarse level triangles between the two adjacent sweep lines which bounds that vertex. To encode a detail of integer m which is not less than 1, we again use a bit string consisting of $m-1$ '1' bits and a '0' bit as a separator.

In Figure 24, all dashed edges are sweep edges. These edges will appear in the final simplified triangle strip as shown in Figure 25. The detail parts from vertex 1 to vertex 5

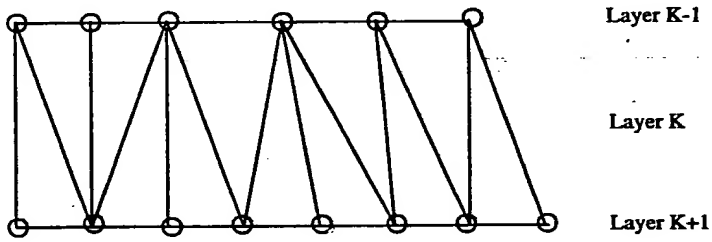


Figure 25: Simplified strip after the inter-layer decomposition

can be encoded as 10, 110, 10, 0, 1110, $\cdots$ and the bit march of the coarse level strip is "011001001010$\cdots$".

We give here a brief complexity analysis of the inter-layer connectivity decomposition. Suppose that three involved contours of two strips have n_0, n_1 and n_2 vertices respectively. Then the two strips have totally about $n_0+2n_1+n_2$ triangles. The exact number may depend on the closeness of the two strips, but the difference will not exceed two. So we regard that number as $n_0+2n_1+n_2$. The cost of bit march string for the coarse level strip will be about n_0+n_2 bits since there exist that number of triangles in the coarse level strip. By the way the details are expressed, the detail cost is equal to the number of triangles in the coarse level strip, that is about n_0+n_2. Therefore, when a mesh is decomposed into a coarser level mesh by the inter-level decomposition. the overall detail cost is roughly equal to the cost for bit march strings for the coarser level mesh which is approximately the number of triangles

in the coarser level. If half of vertices in the fine level are thus decimated, then the number of triangles will be reduced to a half. So the cost of bit march strings is divided into two almost equal parts, one for coarse level bit march strings, one for the details.

Inter-Layer Connectivity Reconstruction Inter-layer connectivity Reconstruction is clear itself. When a vertex layer is added, vertices are added one by one. For each added vertex, the two sweep lines are first decided according to the coarse level bit march string and its associated detail. The two sweep lines decide its boundary. Create an edge to each vertex in the boundary for this added vertex.

All the examples given above are regular triangle strips. Our two-phase decomposition procedure can also be performed on exceptional triangle strips. Bubbles may move forward because of vertex merging. Also the number of special triangle may change because vertex merge may produce degenerated special triangle. This is also true to the second kind of special triangles whose vertices are in the same vertex layer but span more than one contour.

It must be pointed out that it is possible that only one of the two steps is performed in the intra-layer decomposition. We should prohibit performing the intra-layer decomposition for a vertex layer if its number of vertices becomes too small (say three, a regular closed contour has at least three vertices). This feature can be used in view-dependent decomposition.

3.3 Geometry Data Transfer in Progressive Connectivity Transmission

Geometry data in progressive transmission mode will be processed specially. The geometry position of each new vertex in finer level will be encoded by linear prediction with the prediction point determined by a quadratic parametric Bézier curve, as show in Figure 26.

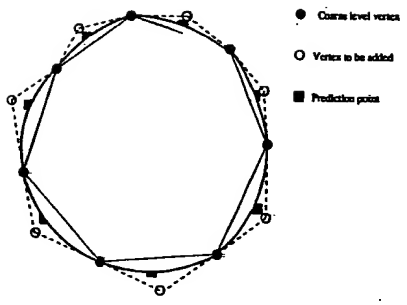


Figure 26: Geometry data transfer

A simple way to choose the prediction point for a new vertex would take the center of the segment with its two adjacent vertex in coarse level as end points. However, prediction in this way will produce large length correction vector which is not desirable for encoding. Instead, we will compute the prediction point from a parametric quadratic Bézier curve, and decide single parameter for all new vertices by solving an optimization problem.

A Bézier curve in Bernstein base with control points $\mathbf{b}_j$ can be written as

$$\mathbf{P}(t) = \sum_{j=0}^{n} \mathbf{b}_j B_{j,n}(t), 0 \leq t \leq 1$$

with Bernstein base function $B_{j,n}(t)$ defined by

$$B_{j,n}(t) = C_n^j t^j (1-t)^{n-j}, \;\; j = 0, 1, \cdots, n$$

and

$$C_n^j = \frac{n!}{j!(n-j)!}$$

Let $\{P_i^k\}$ denote vertices in level k and $\{P_i^{k+1}\}$ vertices in layer k+1 for a contour. Vertex P_i^{k+1} has two level k adjacent vertices P_i^k and P_{i+1}^k, as shown in Figure 27. To determine a quadratic Bézier curve, one only needs to find its three control points $\mathbf{b}_j, j = 0, 1, 2$. In

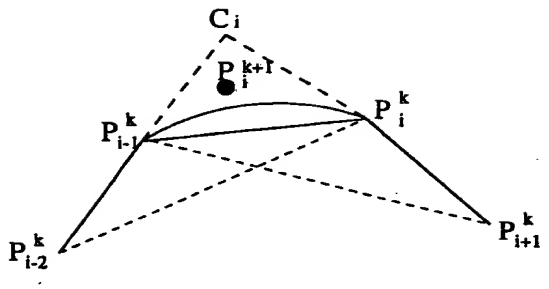


Figure 27: Determination of prediction point

this case, $\mathbf{b_0} = \mathbf{P}^k_{i-1}$, $\mathbf{b_2} = \mathbf{P}^k_i$. To decide the other control point, we first find the plane *pl* where the control points will be located. Plane *pl*'s normal is computed as the normal average of triangle $P^k_{i-2}P^k_{i-1}P^k_i$ and triangle $P^k_{i-1}P^k_iP^k_{i+1}$. Project the two lines $P^k_{i-2}P^k_{i-1}$ and $P^k_{i+1}P^k_i$ onto the plane *pl* and let $\mathbf{C}_i$ be the intersection of the two projected lines. Let $\mathbf{b_1} = \mathbf{C}_i$. Now we have the quadratic Bézier curve

$$\mathbf{P}_i(t) = \mathbf{P}^k_{i-1}B_{0,2}(t) + \mathbf{C}_iB_{1,2}(t) + \mathbf{P}^k_{i+1}B_{2,2}(t), 0 \le t \le 1$$

This quadratic Bézier curve can be regarded as a prediction curve on which a prediction point is chosen. If the nearest point to P^{k+1}_i would be per-suited, one need to pay extra cost by transferring one parameter for each prediction point. The other way is to use $\mathbf{P}_i(0.5)$ as the prediction point and so no parameter needs to be transferred. But the potential problem is that $\mathbf{P}_i(0.5)$ may not be a good prediction point to P^{k+1}_i. A compromising approach would use a single parameter $\tilde{t}$ for all new vertices added to a contour. We take this approach in our algorithm. Mathematically, that single parameter can be computed by solving the following optimization problem

$$\min_{t\in[0,1]} \sum_i ||P_i(t) - P^{k+1}_i||^2_2.$$

Here $||.||_2$ is the Euclidean norm. Finding prediction points in this way produces better results with very small space cost.

For the geometry encoding part of the whole progressive transmission procedure, vertices at the lowest level are encoded by the code difference approach. As resolution level increases, the prediction points for new vertices are first computed. Their corresponding correction vectors are then encoded by PVQ. Bézier curves have been successfully used to locate good prediction points.

4 Tiling Surfaces, Comparison and Experiments

We have described our geometric and connectivity encoding algorithms which are very efficient because they decor-related as much data redundance as possible. This section gives outputs of our implementation. At first, our method is especially efficient for a class of meshes: tiling surfaces. Compression results of various kinds of models are also listed. Finally we present that surface smoothing technique can be utilized in the reconstructed surfaces.

4.1 Tiling Surfaces

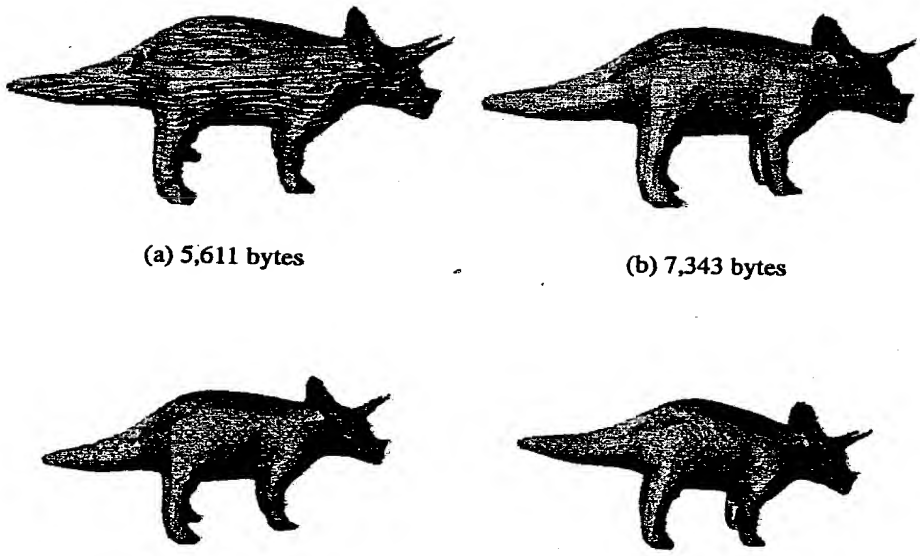
Modern 3D imaging techniques such as computed tomography (CT) and magnetic resonance imaging (MRI) produce a kind of meshed surfaces from some tiling algorithm. Usually, the input to a tiling algorithm is a set of planar cross-sections. The output is a meshed surface which will be called a tiling surface. A tiling surface has the following features. First, all its vertices are naturally grouped by layers, that is, cross-sections. Second, triangles between two adjacent cross-sections form one or more triangle strips where no bubble

could appear. Third, the topology of the input mesh is quite clear since all critical points are easy to grasp.

We find that our compression and encoding method is very suitable for this kind of tiling surfaces. If the first vertex layer in our layering structure are composed of all the vertices in the the first cross-section, all other vertex layers will correspond to other cross-sections. Quite the often in most cases, we can assume that vertices in same vertex layer have the same Z value. It would waste much storage space if encoding the Z value multiple times. All existing encoding scheme can not take advantage of this good feature without changing the algorithm which is supposed to deal with meshes of arbitrary topology. Our algorithm makes use of this feature. It is easy to see that all third components of the code differences are 0. So only one bit is enough to encode each of the third component since the Huffman table for the third component has only one symbol. It is not necessary in our scheme that the planar are parallel to the XOY, or even parallel to each other.

Figure 28 shows the compression results of a object with relative simple topology.

Figure 29 shows the visualization of an entire human skeleton (Freddy). The original tiling surface is constructed from 920 slices, each with 256×256. These 2D images require about $57.5M$ bytes. With some specific threshold value, 2D contours are computed for each of these slices. The final tiling surface is reconstructed from these contours by the algorithm of Bajaj et al. [1]. This tiling surface consists of 285,349 triangles. This object is not simple and has lots of boundary edges. The size of the file containing this tiling surface is $37,896,512$ bytes (each triangle expressed by a triple of coordinates). The compressed files with 15 and 18 target bit rates are of size $346,810$ and $370,505$ bytes respectively. Compression ratios in Figure 29 are computed by dividing the input file size by the size of the files holding the compressed objects. We will soon describe an a more true-worthy way of

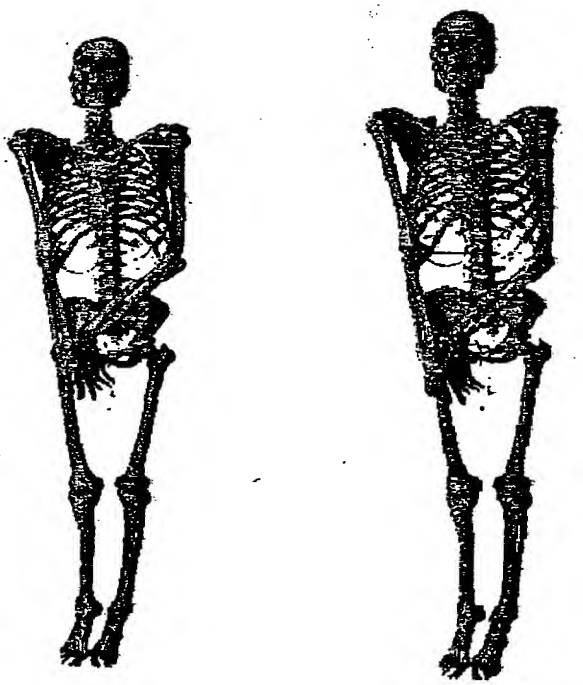


(a) 5,611 bytes (b) 7,343 bytes

(c) 8,487 bytes (d) Original : 337,458 bytes

Figure 28: Tiling surface: simple object

defining compression ratio.



Compression Ratio : 109.3 Compression Ratio : 102.3

Figure 29: Tiling surface: complexity object

4.2 Error Measurements and Algorithm Comparison

In order to compare various geometry compression and encoding methods, it is important to accurately measure the error introduces when substituting the compressed or simplified model S' to the original model S. Any encoding or simplification scheme must derive an

approximation model with good fidelity. To produce a good approximation, at least an upper bound in error must be guaranteed. Basically, there are two kinds of errors: geometric errors and color errors. Geometric errors stand for the pixels deviation between S and S' while color errors express the pixel difference in color when two models are displayed on a screen. Because color errors depend on viewing conditions, so geometric errors are often used in practice. There are two main methods to measure geometric errors: Hausdorff distance and error bound based on vertex displacement.

The Hausdorff distance between two sets A and B is define as

$$H(A,B) = \max(\max_{a\in A}\min_{b\in B}||a-b||_2, max_{b\in B}\min_{a\in A}||a-b||_2)$$

The Hausdorff distance is a symmetric deviation measurement since $H(A,B) = H(B,A)$. Also if $H(A,B) = 0$ then A and B are identical set. However, small $H(A,B)$ does not mean the shapes of A and B are similar. In fact, they could be differ significantly. The cost of computing Hausdorff is high if the two model are very large. The computation of Hausdorff distance can be performed as follows. First compute the intersection of the skeleton of one polyhedron with the boundary of the other and vice versa. The intersection should contains points in the relative interior of the faces of the first polyhedron when the Hausdorff distance is realized. The skeleton of a polyhedron S is the set points whose shortest path to the boundary of S has at least three distance vectors.

Suppose that A and B are two triangle mesh models. The error bound based on vertex displacement is defined as

$$E(A,B) = \max_{v\in A, v'\in B}||v-v'||_2$$

There should a one-to-one correspondence between vertices of A and those of B. Since triangles are linear convex combinations of vertices, if all vertices of a triangle of A are dis-

placed within a fixed distance in B, then no interior point in that triangle will be displaces at a distance exceeding that fixed distance. This error bound is often used to estimated deviation between a compressed models and the original since the one-to-one vertex correspondence is usually quite clear (see IBM scheme). This kind of error bound is easy to compute and the cost is much lower than that of computing Hausdorff distance.

The error metric used throughout this thesis for single-resolution compression experiment is as follow. Model A is the input mesh to the compression/encoding process (undistorted). Model B is produced by the decoding process. A and B have the exact same connectivity but different vertex position. For each vertex of A, the closest vertex of B is determined and the distance is reported. For each vertex of B the closest vertex in A is determined and the distance is reported. A histogram of the (weighted) distances is computed, and the minimum, maximum, mean, and standard deviation of distances is reported. The evaluation is symmetric with respect to A and B. Both model A and B are normalized such that either one of the following applies: (a) the bounding box diameter of the first model A is normalized to 100 or (b) the bounding box diameter that is the largest among A and B is normalized to 100.

Taubin and Rossignac's topological surgery [17] first builds vertex layers by the layering decomposition scheme. Its objective is to minimize the total number of runs of the vertex spanning tree and its dual triangle tree. In order to make a triangle strip long, contours are converted (called rings in [17]) into a spiral so that vertex runs (and hopefully triangle runs) are as long as possible. Good compression results have been reported for general meshes. To cope with triangular manifold such as Euler characteristic other than 2, non-orient-able, and with boundaries, their algorithms need to be extended by introducing new concepts like "jump edges".

Our compression and decompression algorithms are also based on layered decomposition but we have used it in a different way. Although our idea of layering decomposition originally comes from designing algorithms for tiling surfaces compression and encoding, the concept of layering decomposition was definitely first used in [17]. In our algorithm, contours will not be merged to form spirals since no vertex tree needs to be constructed and thus encoded. It looks like that the average length of our triangles strips is shorter. However, for many large mesh models, our algorithms even produce better average strip length. The phone model below may explain this. Our method avoids constructing vertex spanning tree and triangle tree, which is an advantage. Its disadvantage is the need to encode special triangles.

Another advantage of our method is its ability to deal with both manifold and non-manifold meshes in a uniform and combined way. Any generalized triangle strip is associated with its child contour. There may exist more one strips associated with a specific contour. Any special triangle, whose vertices are located in as least two contours in the same layer, is encoded by a triple of local indices. Luckily, these kinds of special triangles for most models only occupy a very small portion of the total triangles. This can be seen from our empirical results below.

This main advantage of the method in this paper may lie in its ability of building multiresolution meshes. We have taken two types of simplification procedures to build the hierarchy of the originally mesh. This kind of hierarchy makes progressive connectivity transmission possible.

Progressive simplicial complex (PSC) represents a given mesh model as a coarse model along with a sequence of refinement transformations [14]. The PSC allows the input model to be an arbitrary triangulation, and the coarsest level to be a single vertex. It also encodes

the geometry and topology by the sequence of refinement transformations, which in fact define a multiresolution framework. PSC supports progressive transmission and defines a continuous sequence of approximating models. It also uses linear prediction to encode geometry, quite similar to [3]. For an average 2-manifold of n vertices, connectivity encoding requires about $O(nlog_2n+7)$ bits. The overall compression ratio listed in [14] is about 6.0 which is not good enough. Another disadvantage is its high construction time cost.

4.3 Experiments

Table 5 presents the overall compression and encoding result with geometric coding cost about 13 bits per vertex. In this table, the column "Original" lists the sizes of original VRML files which store the input mesh models. "Gzipped", gives the sizes of compressed VRML file with the "gzip' compressor available on most UNIX platform. "Compressed" stands for the compressed file sizes with our single-resolution compression and encoding scheme. All the files are recorded in bytes. The compression ratios (CR) are computed from the data in the third and fifth columns. The geometric is the mean error of all vertex deviations.

We give here a set of pictures from our experiment. The original models are of various kinds. Their sizes change from hundreds to hundreds thousands faces. They could be simple meshes or meshes of high Euler characteristics. They also could be composed of several components, each of which is a connected mesh. The geometric errors included are the mean errors.

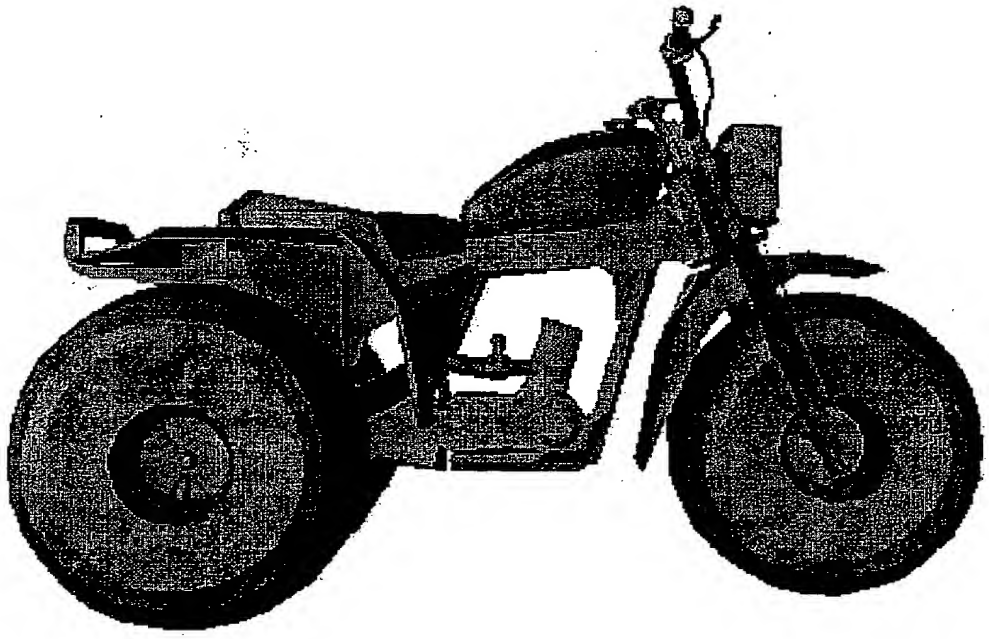


Figure 30: Honda (Compressed, 17,513 bytes, error = 0.108)

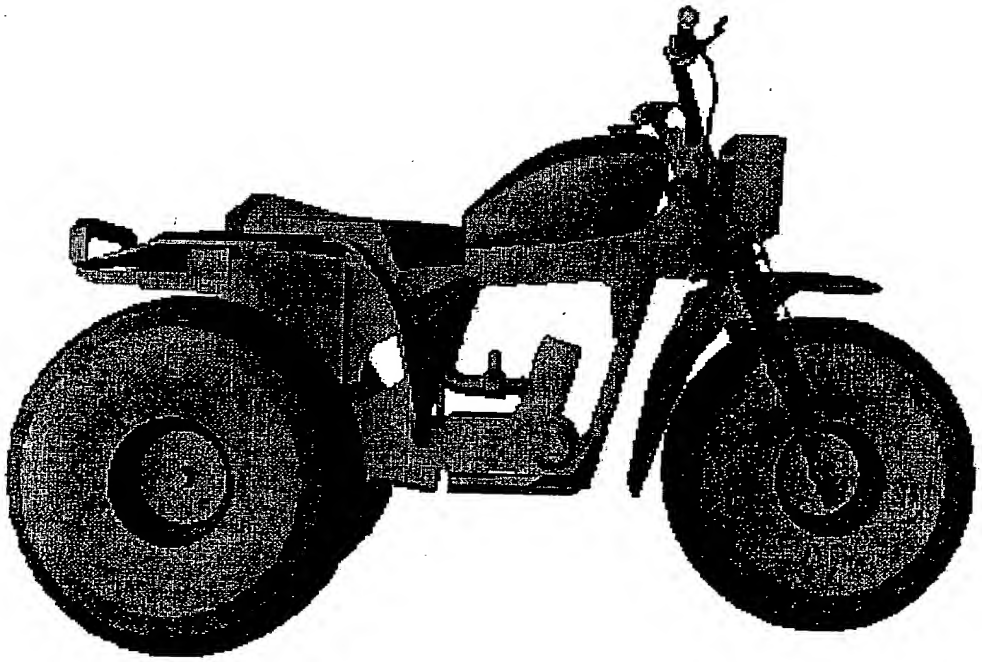


Figure 31: Honda (Compressed, 22,389 bytes, error = 0.046)

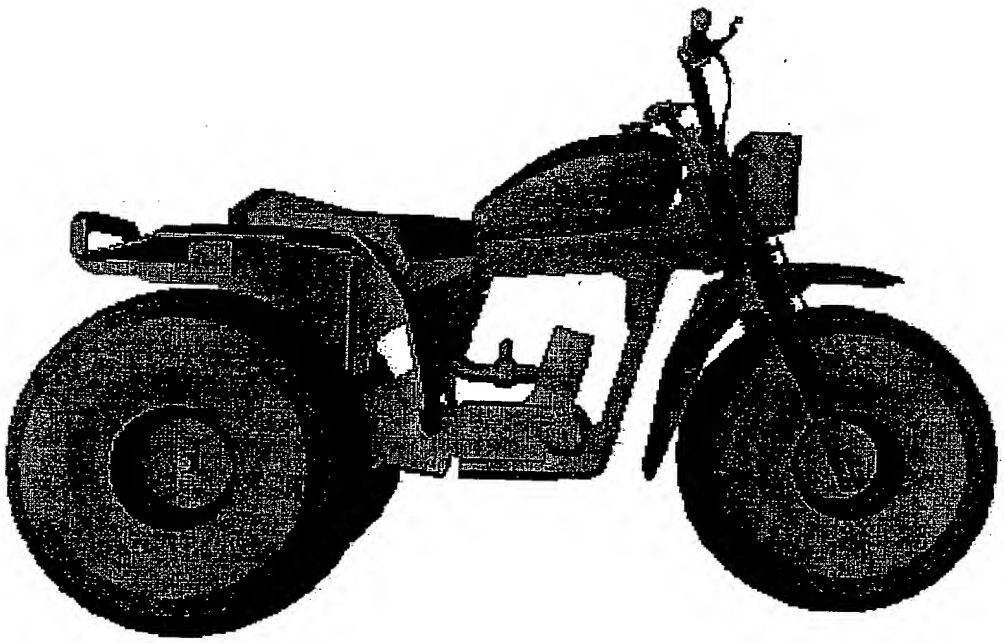


Figure 32: Honda (Original, 483,880 bytes, 13,594 triangles)

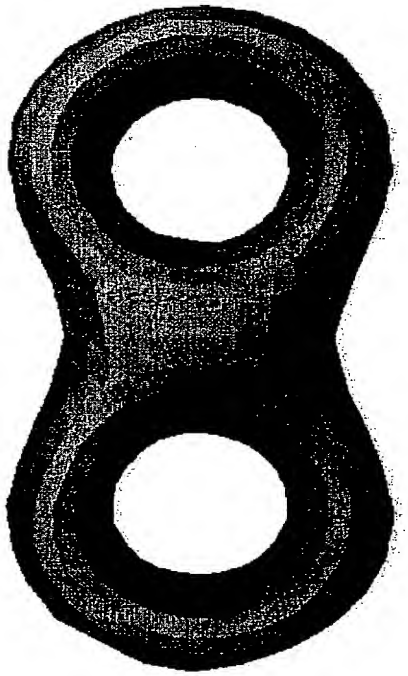


Figure 33: Eight (Compressed, 1,715 bytes, error = 0.343)

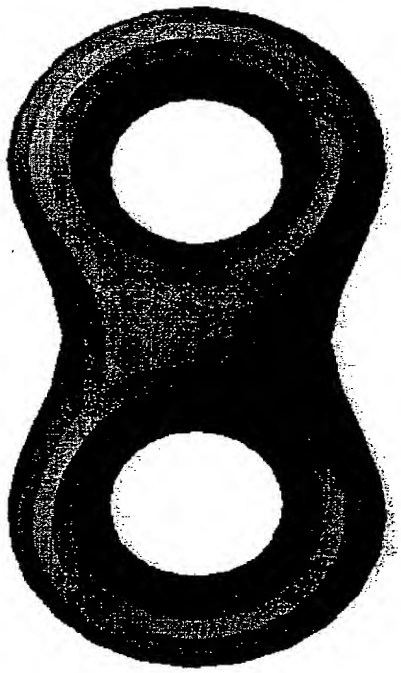


Figure 34: Eight (Compressed, 2,372 bytes, error = 0.083)

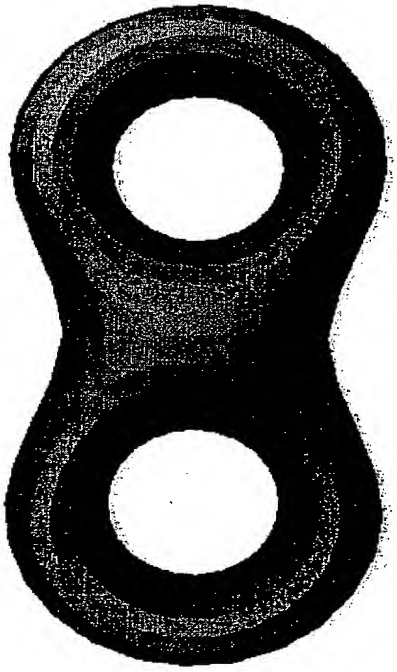


Figure 35: Eight (Original, 49,551 bytes, 1,536 triangles)

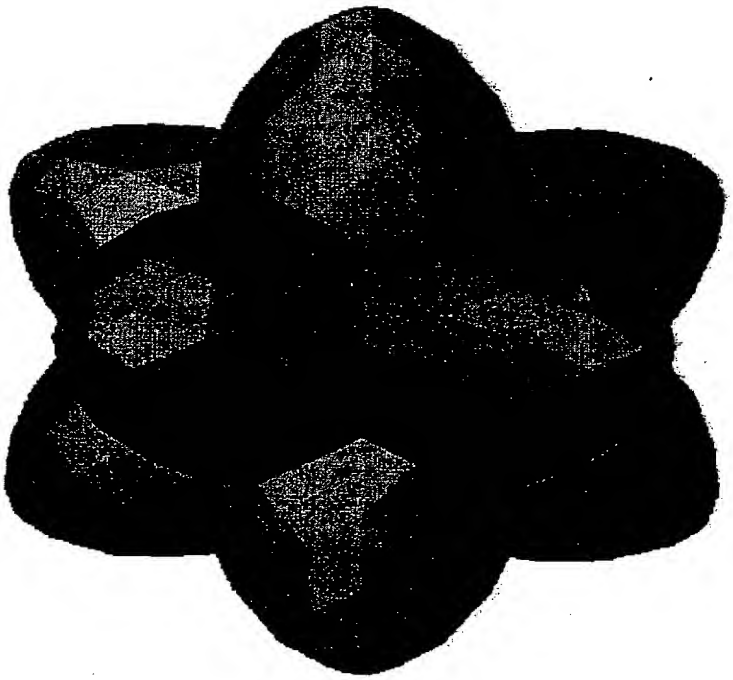


Figure 36: Shape (Compressed, 4,277 bytes, error = 0.385)

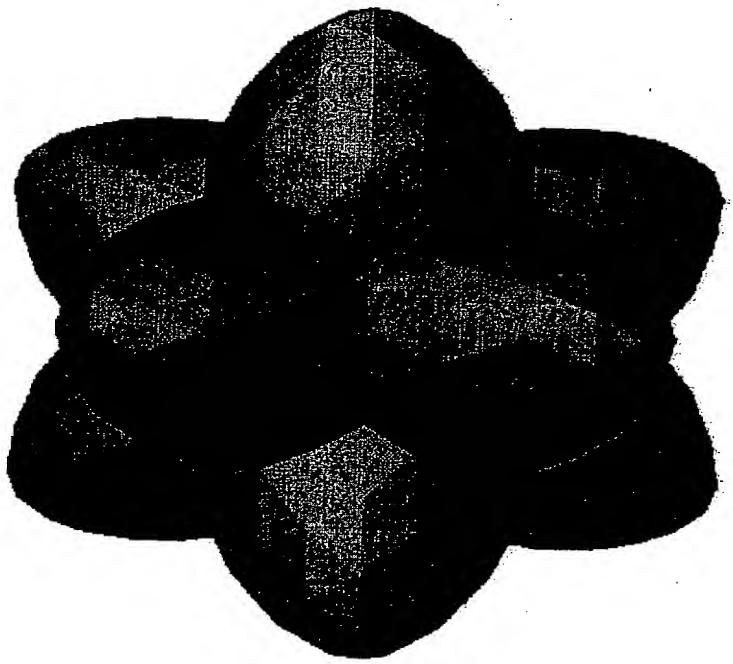


Figure 37: Shape (Compressed, 6,627 bytes, error = 0.104)

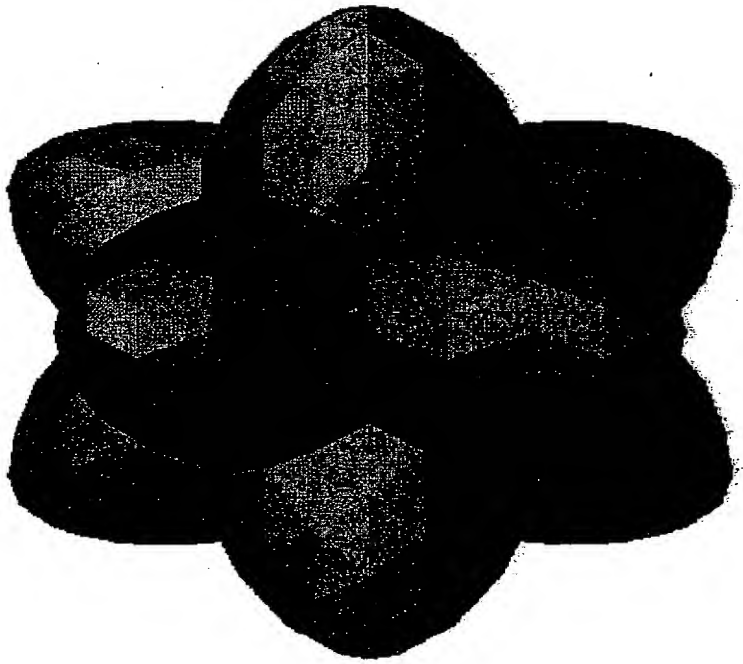


Figure 38: Shape (Original, 159,151 bytes, 5,120 triangles)

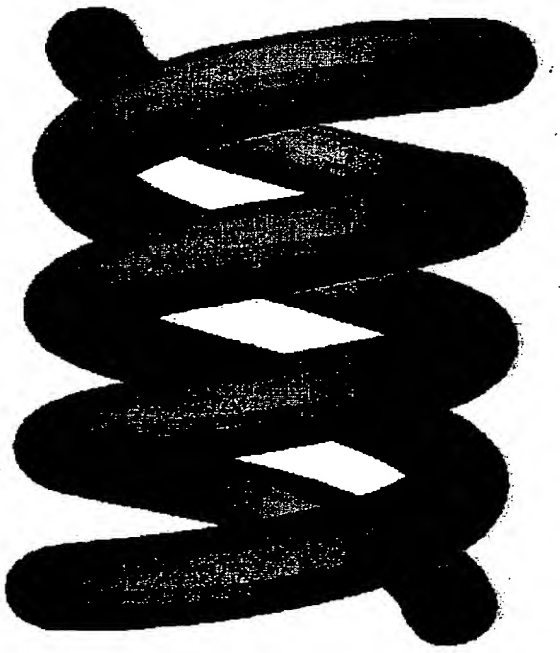


Figure 39: Helix (Compressed, 6,126 bytes, error = 0.002)

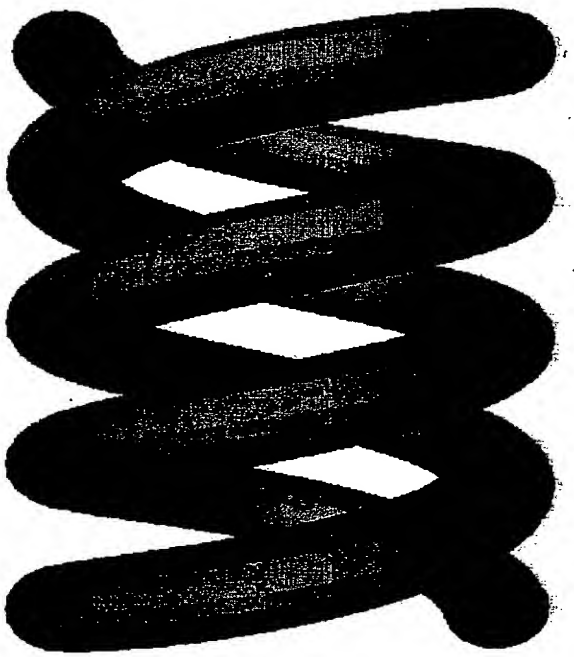


Figure 40: Helix (Compressed, 8,970 bytes, error = 0.002)

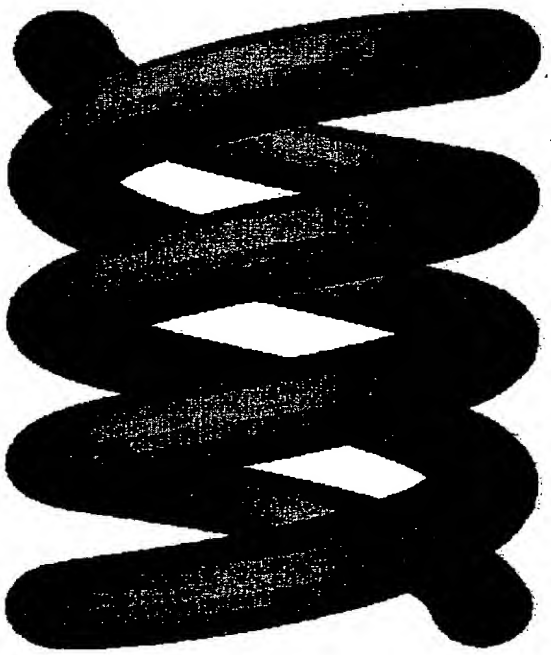


Figure 41: Helix (Original, 311,453 bytes, 6,400 triangles)

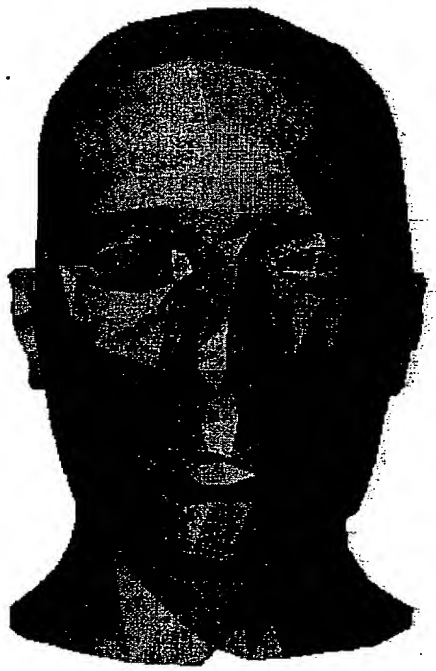


Figure 42: Mannequin (Compressed, 1,902 bytes, error = 0.446)

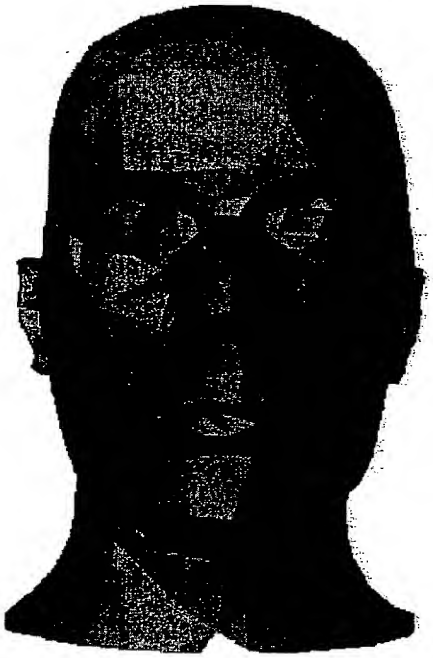


Figure 43: Mannequin (Compressed, 2,493 bytes, error = 0.120)

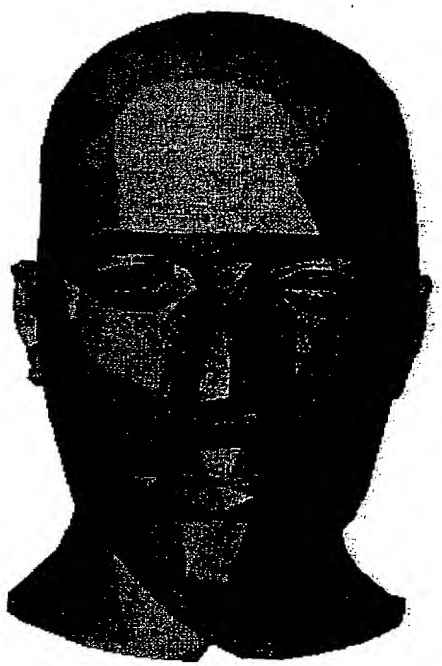


Figure 44: Mannequin (Original, 39,967 bytes, 1,355 triangles)

model	nV	Original	Gzipped	Compressed	CR	error
bunny	34835	3415624	803297	63843	53	0.314
fandisk	6475	627975	142600	13024	48	0.225
head	3690	281540	47685	8578	32	0.253
horse	11135	1087962	273324	23718	45	0.090
mannequin	689	39967	17285	1675	23	0.920
mechpart	1932	109323	39799	4081	26	0.318
phone	83044	8558332	2223880	157983	54	0.185
skull	84635	8622253	1753419	243019	35	0.218
trice	4986	337458	81852	13690	24	0.225

Table 5: geometric coding

5 Conclusion and Further Work

Our topological layering scheme coupled with vector quantization, has provides a novel combination of compressing and encoding of both the topology (connectivity) and geometry (vertex coordinates) of arbitrary polygon meshes. Its superiority over prior methods stems from its capability of handling arbitrary polygon meshes (high genus,non-manifold), achieve higher compression rates, as well as allows for progressive transmission of the encoded geometry with error/distortion control parameters. There are several future research direction. In our scheme, the compression ratio are largely determined by the total numbers of contours, the average length of triangle strips and the number of special triangles. Good results can be expected with small number of contours, large average length of triangle strips and a smaller number of special triangles. As regard to entropy coding, on one hand, it would be desirable if the input's entropy is reduced as small as possible; on the other hand,

Figure 45: Skull (Compressed, 28,595 bytes, error = 0.188)

Figure 46: Skull (Compressed, 47,759 bytes, error = 0.078)

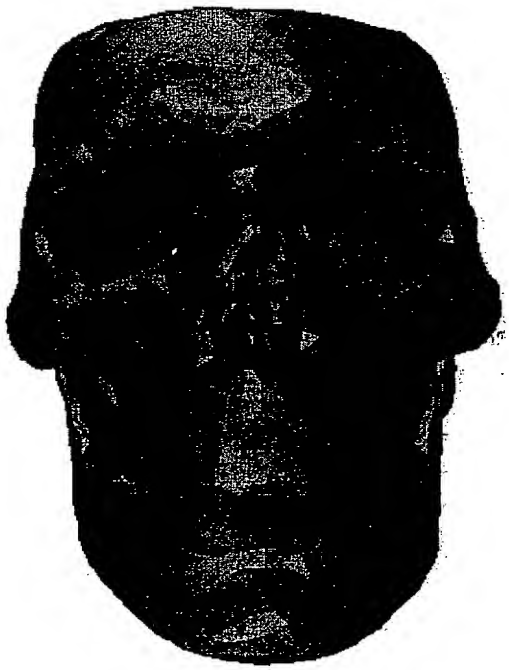


Figure 47: Skull (Original, 741,547 bytes, 22,104 triangles)

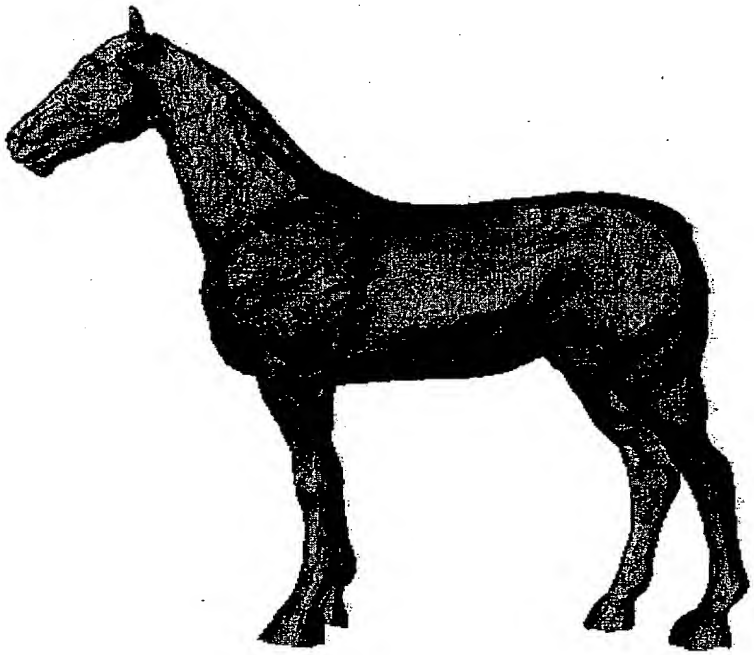


Figure 48: Horse (Compressed, 23,618 bytes, error = 0.090)

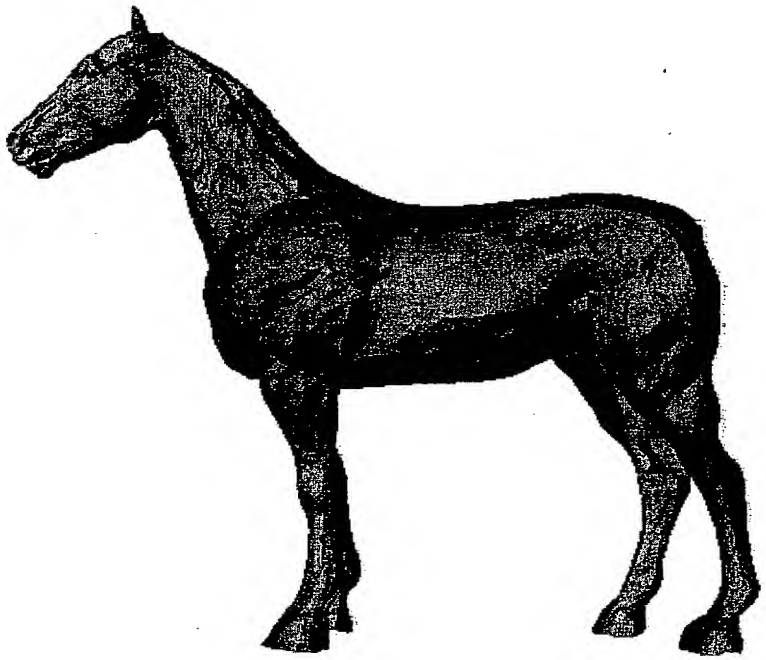


Figure 49: Horse (Compressed, 32,973 bytes, error = 0.033)

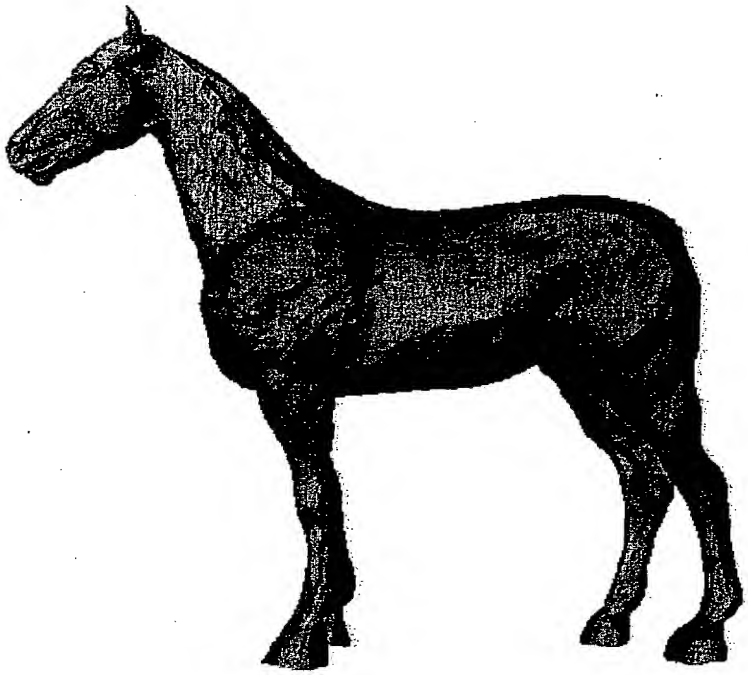


Figure 50: Horse (Original, 1,087,962 bytes, 22,246 triangles)

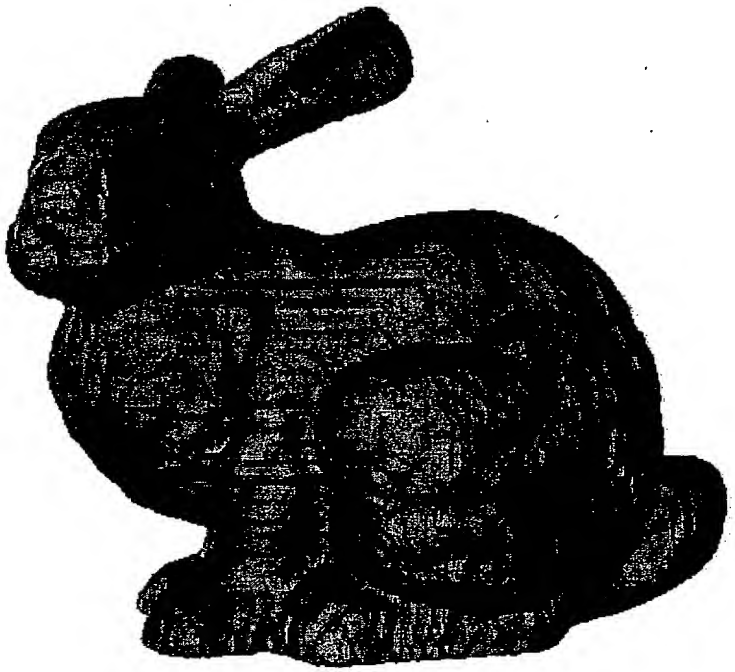


Figure 51: Bunny (Compressed, 64,843 bytes, error = 0.315)

Figure 52: Bunny (Compressed, 85,092 bytes, error = 0.189)

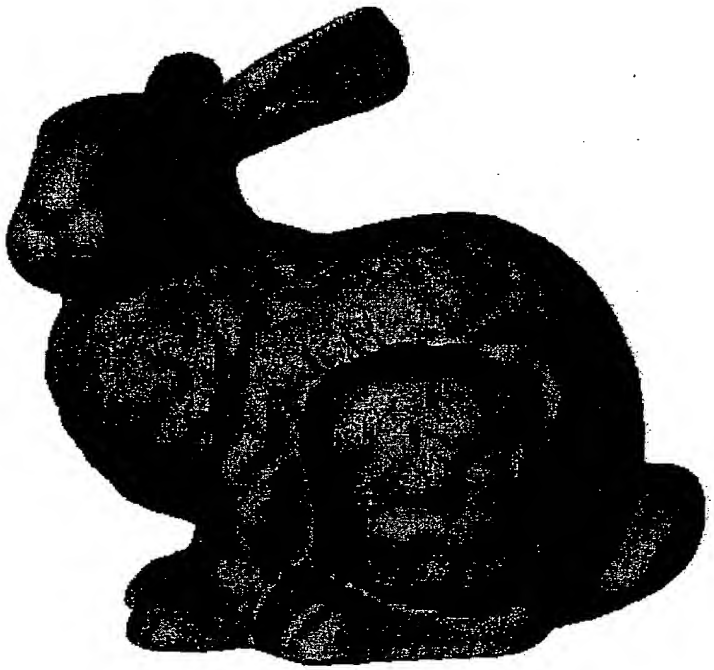


Figure 53: Bunny (Original, 2,268,236 bytes, 69,473 triangles)

Figure 54: Phone (Compressed, 139,869 bytes, error = 0.315)

Figure 55: Phone (Compressed, 190,966 bytes, error = 0.139)

Figure 56: Phone (Original, 8,558,332 bytes, 165,963 triangles)

coding method other than Huffman coding should be designed so the real cost per symbol should be as near to the entropy as possible (context based arithmetic coding may be the one). The second is to encode and transmit progressively the connectivity information for dynamically simplified polygonal mesh scenes. Connectivity progressive transmission needs to proceed in a more feasible way while mesh decomposition is feature-based. We also need to find the best way of combining progressive transmission and view-dependent refinement of 3D geometry. Relation between wavelet-based approaches and polygonal surface based approaches for progressive transmission of 3D geometry needs to be explored so that the better one can be taken. Another issue is related to data transfer over network. Treating lost packets during transmission and adjust resolution according to transmission bandwidth and actual network throughput. Compression and encoding of tetrahedral meshes are also important for practical purpose. Another avenue to explore is the compression and encoding of arbitrary spline meshes for NURB models of large CAD objects.

6 Appendix

Data Structures

```
typedef float Position[3];
typedef unsigned int Triangle[3];
typedef struct {
    int nl;
    Layer **layers;
} TTree;
typedef struct {
    int nc;
```

```
    int nb;
    int *branch;
    Contour **contours;
    int niv;
    int *iv;
    int ntri;
    int *tri;
} Layer;

typedef struct {
    int depth;
    int closed;
    int nv;
    int *verts;
    int startB;
    int endB;
    int nstrip;
    Strip *strips;
} Contour;

typedef struct {
    int parent;
    int pstart;
    int cstart;
    int direct;
```

```
    char closed;
    int len;
    char *march;
    int nb;
    Bubble *bubbles;
} Strip;

typedef struct {
    int cstart;
    int pstart;
    int pspan;
    int ntri;
    Triangle *tri;
} Bubble;
```

The Syntax of Data Stream Table 6 shows the format of overall data stream of a compressed mesh.

HEADER	GEOMETRY	CONNECTIVITY

Table 6: Data stream format

The two fields $GEOMETRY$ and $CONNECTIVITY$ give geometry and connectivity information respectively. We will illustrate them later. The field listed as $HEADER$ is a fixed length field which contains information that decoder needs to recovery the encoded geometry and connectivity. Table 7 displays all the details.

LENGTH	BITS_NC	BITS_NIV	BITS_LEN
BITS_NB	BITS_NV	BITS_NSTRIP	BITS_PARENT
BITS_PSTART	BITS_CSTART	BITS_NBUB	BITS_BUB_PSTART
BITS_BUB_CSTART	BITS_BUB_PSPAN	BITS_BUB_NTRI	BITS_NTRI
BITS_VERT	BITS_R	BITS_PHI	BITS_THETA

Table 7: The header in the data stream format

The field $LENGTH$ is the total length of the compressed data stream. This length value assists the decoder to allocate buffer for incoming data. The left fields of Table 7 contain values which indicate how many bits are used to encode related values in the bit stream. To encode these constants, currently 5 bits are used for each of them.

$BITS\_NC$ specifies the number of bits to encode the number of contours in each vertex layer. Suppose that $max\_contours$ is the maximum number of contours among all vertex layers. Let NC be the smallest positive integer such that $2^{NC} > max\_contours$. Then the fields $BITS\_NC$ contains the value NC. The other constants have similar explanation which will be omitted. The $BITS\_LEN$ field gives the number of bits to encode the length of a generalized triangle strip while $BITS\_NSTRIP$ bits are used to show the number of strips associated with a single contour. The $BITS\_NV$ field is for encoding the number of vertices in each contour. The $BITS\_NIV$ field gives the number of bits which are needed to encode all isolated vertices in a layer. Here an isolated vertex can be regarded as a contour with only one vertex. But for efficiency reason, it will not be taken as a contour. Branch vertices are specially encoded because they will be used in more than one contour. The $BITS\_NB$ field contains the number of bits to encode the number of branching vertices in a vertex layer.

Each generalized triangle strip involves two contours in adjacent layers. We call the contour in the smaller numbered layer as parent contour and the other as child contour. Accordingly, the layer containing the parent contour is called the parent layer while the layer containing the child contour is called the child layer. As shown in our data structure, all strips are associated with their child contours. Since the number of strips which are associated with a (child) contour may be greater than one, both the local index of the starting vertex in the parent contour and the local index of the starting vertex in the child contour must be encoded (by $BITS\_PSTART$ and $BITS\_CSTART$ bits). Also the decoder must know the parent contour number because the parent layer may be composed of more than one contour. Recall that contours in each layer are numbered with consecutive integers, starting from 0.

To encode a bubble in a generalized triangle strip, one must show the starting vertex in the parent contour ($BITS\_BUB\_PSTART$) and the starting vertex in the child contour ($BITS\_BUB\_CSTART$). Also the vertex span in the parent contour for that bubble must be specified ($BITS\_BUB\_PSPAN$). For each generalized strip, the number of bubbles must also be put in the data stream ($BITS\_NBUB$).

The value in the $BITS\_NTRI$ field is the number of bits to encode the number of special triangles. A special triangle spans at least two contours and all its three vertices are located in the same layer. The $BITS\_VERTS$ field value shows the number of bits which is large enough to encode local indices of vertices of any special triangle. In fact, this value is the smallest integer whose power of two is greater than the vertex number in any layer.

The three fields $BITS\_R$, $BITS\_PHI$, and $BITS\_THETA$ contain the numbers of bits to encode r, ϕ and θ. The decoder will use them to build Huffman tables. They are not directly used for every difference vector in our geometry encoding scheme.

The $GEOMETRY$ field in Table 6 contains geometry information. Since in our encod-

ing scheme, vertices are encoded in strict order. And no vertex spanning tree is used. Table 8 shows the whole geometry part in the data bitstream.

RESOLUTION	FACTORX
FACTORY	FACTORZ
NUM_LAYERS	
GEOMETRY DATA FOR THE FIRST LAYER	
GEOMETRY DATA FOR THE SECOND LAYER	
⋮	
GEOMETRY DATA FOR THE LAST LAYER	

Table 8: The format of encoded geometry data

The first field $RESOLUTION$ in Table 8 specifies the number of bits used to fully encode a float number. For example, the coordinate of the starting vertex in each vertex contour usually needs to be fully encoded. In most cases, 16-bit float resolution is good enough. We keep this field so that it can be adjusted by various applications. The next three fields $FACTORX, FACTORY$ and $FACTORZ$ are used for normalization purpose. The NUM_LAYERS field gives the total number of layers in the decomposed mesh. Finally, the geometry for each layer is encoded successively.

For the geometry of each layer, Table 9 sketches its bit stream format.

The $NUM\_CONTOURS$ field gives the number of contours in that layer. It must be noticed that we all branching vertices and isolated vertices are not taken as single-vertex contours for the efficiency reason. The third field contains possible branching information. It depends on the value in the second field, $BFLAG$ which is of 1 bit length. If $BFLAG$ contains 0, there is no branching vertex in this layer and thus the third field in Table ?? is

NUM_CONTOURS	
BFLAG	DATA BATCHING VERTICES
GEOMETRY DATA FOR THE FIRST CONTOUR	
GEOMETRY DATA FOR THE SECOND CONTOUR	
⋮	
GEOMETRY DATA FOR THE LAST CONTOUR	
DATA FOR ISOLATED VERTICES	

Table 9: The format of geometry data for each layer

empty.

The geometry of each contour is encoded separately. Only the geometry of non-branching vertices in a contour is encoded. Recall the fact that every branching vertex appears in more than one contour. So, in order to avoid encoding a branch vertex multiple times, we encode every branch vertex separately, independent of any contour it is located.

The data for the branching vertex is shown in Table 10.

NUMBER BATCHING VERTICES
GEOMETRY DATA FOR THE FIRST BRANCHING VERTICES
GEOMETRY DATA FOR THE SECOND BRANCHING VERTICES
⋮
GEOMETRY DATA FOR THE LAST BRANCHING VERTICES

Table 10: The format for branching vertices

The first field in Table 10 gives the total number of branching vertices. Then these vertices are encoding one by one. The geometry data for each branching vertices is the coordinate

whose three float components are encoded each with $RESOLUTION$ bits.

The last field of Table 9 is related to isolated vertices. Table 11 gives the bit stream format for the possibly existing isolated vertices.

NUMBER BATCHING VERTICES
GEOMETRY DATA FOR THE FIRST ISOLATED VERTICES
GEOMETRY DATA FOR THE SECOND ISOLATED VERTICES
⋮
GEOMETRY DATA FOR THE LAST ISOLATED VERTICES

Table 11: The format for branching vertices

The first field in Table 11 is encoded by the number of bits $BITS\_NIV$. Usually, isolated vertices seldom occur. If they do, the number of isolated vertices is often very small. In most of our experiments, $BITS\_NIV$ is 0, 1 or 2. If the decoder knows that $BITS\_NIV$ is 0, it will neglect recovering this isolated vertices because there is none. If there do exist isolated vertices, then the geometry encoding is quite the same as that for branching vertices.

It must be stressed here that the total number of both branching and isolated vertices is usually so small that it is less than one percent of the total number of vertices of the mesh. So these special vertices will not significantly affect the final compression ratio although it is expensive to encode each of them (considering the cost to encoding vertices by prediction).

The meaning of the first field of Table 12 is self-explained. Both the second and the fourth fields are of 1-bit length. In case the second field contains 0 value, the third field will be empty. If it is 1, the contour starts with a branching vertex. The third field then contains the local index of the first vertex To repeat, all branching vertices in a layer are increasingly numbered starting from 0. The Local index indicates which is the starting vertex of

NUMBER OF VERTICES	
FFLAG	FINDEX
LFLAG	LINDEX
GEOMETRY DATA FOR THE FIRST VERTIES	
GEOMETRY DATA FOR THE ADJUSTED CONTOUR	

Table 12: The format of the geometry data for a single contour

the current contour. The $FINDEX$ field is of length $BITS\_NB$ which is a small number. If global index were used, more bits would be needed for each branching vertex. The $LFLAG$ and $LINDEX$ fields show whether the last vertex in the contour is a branching vertex. The explanation about the first vertex can be directly applied to this last vertex case. After excluding the first vertex and possibly the last vertex (if it is a branching vertex), we get an adjusted contour. If the value in the $FFLAG$ is 1, then the sixth field of Table 12 for the geometry of the first vertex in the contour is empty because in that case the first vertex is a branching one and it has already been encoded. Otherwise, the coordinate of the first vertex is fully encoded, each of its three float components by $RESULOTION$ bits. The following table shows the bit stream format of the geometry of an adjusted contour.

CODES FOR THE FIRST DIFFERENCE VECTOR
CODE DIFFERENCE FOR THE SECOND DIFFERENCE VECTOR
CODE DIFFERENCE FOR THE THIRD DIFFERENCE VECTOR
⋮
CODE DIFFERENCE FOR THE LAST DIFFERENCE VECTOR

Table 13: The format of the geometry data for a single contour

The difference vector is a vector which is the difference between two adjacent vertex coordinates. The three integer codes for the first difference vector are directly encoded by $BITS\_R$, $BITS\_PHI$ and $BITS\_THETA$ bits respectively. Let (r_i, ϕ_i, θ_i) be the triple of integer codes for the i^{th} difference vector. Then the code difference for the i^{th} difference vector is defined as $((r_i - r_{i-1} + N_r)\%N_r, (\phi_i - \phi_{i-1} + N_\phi)\%N_\phi), (\theta_i - \theta_{i-1} + N_\theta)\%N_\theta))$ with the symbol % being a modular operation. So the code difference for each difference vector consists of three positive integers within the intervals $[0, N_r - 1]$, $[0, N_\phi - 1]$ and $[0, N_\theta - 1]$ respectively. For simiplicity, we use $(\Delta r_i, \Delta phi_i, \Delta\theta_i)$ to denote the code difference. Remember, from now on, that $Deltar_i = (r_i - r_{i-1} + N_r)\%N_r$ and it is a nonnegative integer. There is a very good property about the distribution of the components of code differences. We notice that most $\Delta r's$ accumulate around the two endpoints of the interval $[0, N_r - 1]$. $\Delta\phi$ and $\Delta\theta$ have the similar property. We can make use of this nice property to design a very efficient geometry encoding algorithm. The idea is to regard each component of the code difference as an integer symbol. Statistics algorithms such as Huffmann coding and arithmetic coding can then be used to encode these symbols. In those algorithm, the most frequently occurring symbol is assigned a code of the fewest bits. On contrary, the least frequently occurring symbol is assigned a code of the most bits.

The bit stream format of the connectivity data is quite straightforward (see Table 14).

CONNECTIVITY DATA FOR THE FIRST LAYER
CONNECTIVITY DATA FOR THE SECOND LAYER
⋮
CONNECTIVITY DATA FOR THE LAST LAYER

Table 14: The bitstream format of the connectivity data

The connectivity of each layer will be encoded into a bit stream with the format of Table 15.

CONNECTIVITY DATA ASSOCIATED WITH THE FIRST CONTOUR
CONNECTIVITY DATA ASSOCIATED WITH THE SECOND CONTOUR
⋮
CONNECTIVITY DATA ASSOCIATED WITH THE LAST CONTOUR

Table 15: The bitstream format of the connectivity data for a single layer

And all generalized triangle strips associated with a single contour are encoded by the format shown in the following table (Table tab:conn-contour).

NUMBER OF STRIPS
CONNECTIVITY DATA FOR THE FIRST STRIP
CONNECTIVITY DATA FOR THE SECOND STRIP
⋮
CONNECTIVITY DATA FOR THE LAST STRIP

Table 16: The bit stream format of the connectivity data for a single contour

Table 17 gives the format to encode a single generalized triangle strip.

PARENT	PSTART
CSTART	LENGTH
DATA FOR BIT MARCH	

Table 17: The bit stream format of the connectivity data for a single strip

The $PARENT$ shows which is the parent contour of the current strip. The $PSTART$ and $CSTART$ fields contain the local indices of the starting vertices. The $LENGTH$ field

has the value which is the length of the bit march. The last field is encoded by the Huffman coding. The symbols are obtained by simply grouping four consecutive marching bits.

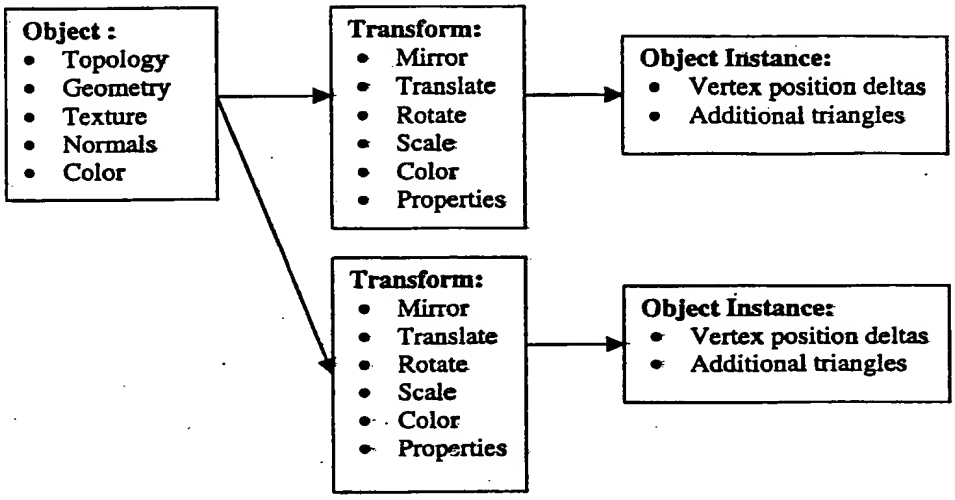
Geometry Delta Coding for Nearly Symmetric Objects

VRML and MPEG-4 BIFS allow copies of individual 3D objects to be placed in a scene after spatial or property transformation. For example, a forest of trees can be composed of one reference tree which is translated and rotated many times. The communication of this forest scene is very efficient because only one tree needs to be transmitted. However, many natural and polygonal objects are composed of repeated instances of component parts which have undergone one or more transformations and small modifications. In these cases, the modifications to each instance of component part are costly to transmit using currently available graphics description standards. For example, a human head and body exhibits left-right symmetry with the exception of small (typically) asymmetries. The size of a human body model can be reduced by a factor of two by replacing one side by a reference to a mirrored instance of the other side. If the two sides are similar enough to share the same topology (vertex connectivity and number), the differences between them can be limited to differences in geometry (vertex positions). Other examples of 3D models with symmetric or repeated components are:

- machines and parts
- biological structures (organs, plants, animals)
- vehicles
- architecture
- furniture
- appliances
- clothing
- musical instruments
- celestial bodies

Encoding the Geometry Deltas

A generalized data structure is shown in Figure 1. A given **Object** is coded as described above in this paper. The placement of **Instances** of the **Object** in the scene is done using a **VRML or MPEG-4 BIFS Transform**. The geometry (vertex position) deltas are applied to the vertices of each instanced object assuming that the topology is unchanged from the reference (transmitted) object. The geometry deltas are coded using the same algorithms and bitstream syntax used for the reference object geometry as described above in this paper. The geometry coding algorithm allows the vertex position resolution to be specified. In the case of geometry delta coding, the resolution is usually low allowing fewer possible delta values and a corresponding reduction in bitrate. Also, most geometry deltas are typically zeros which can be efficiently coding by selecting a variable length code for zero of 1 bit. Additional triangles can also be added using existing VRML or MPEG-4 BIFS syntax.



References

[1] BAJAJ, C. L., COYLE, E. J., AND LIN, K. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Procession 58*, 6 (1996), 524–543.

[2] CHOW, M. M. Optimized geometry compression for real-time rendering. In *Proceedings of IEEE Visualization '97* (Phoenix, AZ, 1997), pp. 347–354.

[3] DEERING, M. Geometric compression. *Computer Graphics (Proc. SIGGRAPH)* (1995), 13–20.

[4] FUCHS, H., KEDEM, Z. M., AND USELTON, S. P. Optimal surface reconstruction from planar contours. *Communications of the ACM 20* (1977), 693–702.

[5] GERSHO, A., AND GRAY, R. *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer Academic Publishers, 1992.

[6] GRAY, R. M. Vector quantization. *IEEE ASSP Magazine 1*, 2 (1984), 4–29.

[7] HANG, H.-M., AND WOODS, J. W. *Handbook of Visual Communications*. Academic Press, 1995.

[8] HOPPE, H. View-dependent refinement of progressive meshes. *Computer Graphics (Proc. SIGGRAPH '97)* (1997), 189–198.

[9] HUFFMAN, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE 40* (1952), 1098–1101.

[10] LEUBKE, D., AND ERIKSON, C. View-dependent simplification of arbitrary polygonal environments. *Computer Graphics (Proc. SIGGRAPH '97)* (1997), 199–208.

[11] LI, J., LI, J., AND KUO, C.-C. J. Progressive compression of 3d graphic models. In *IEEE Proceedings of Multimedia Computing and Systems* (Ottawa, Canada, 1997).

[12] LUEBKE, D. Hierarchical structures for dynamic polygonal simplification. Tech. Rep. TR 96-006, Dept. of Computer Science, University of North Carolina at Chapel Hill, 1996.

[13] MEYERS, D., SKINNER, S., AND SLOAN, K. Surfaces from contours. *ACM Trans. on Graphics 11*, 3 (1992), 228–258.

[14] POPOVIC, J., AND HOPPE, H. Progressive simplicial complexes. *Computer Graphics (Proc. SIGGRAPH)* (1997), 217–224.

[15] ROSSIGNAC, J. R. Simplification and compression of 3d scences. In *Tutorial Eurographics '97* (1997).

[16] ROSSIGNAC, J. R., AND BORREL, P. Multiresolution 3d approximations for rendering complex scences. Tech. Rep. RC-17697, IBM T.J. Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, 1992.

[17] TAUBIN, G., AND ROSSIGNAC, J. Geometric compression through topological surgery. Tech. Rep. RC-20340, IBM T.J. Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, 1996.

[18] VARSHNEY, A. *Hierarchical Geometric Approximations*. PhD thesis, Department of Computer Science, University of North Carolina, 1994.

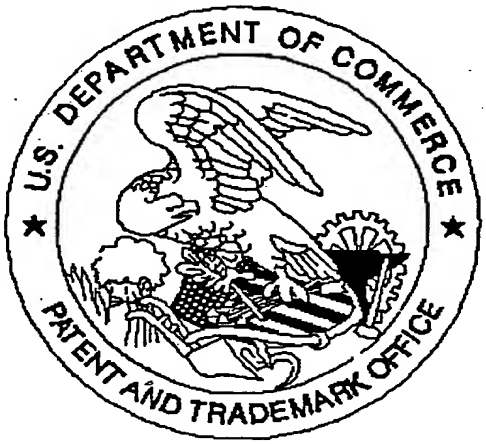
[19] WITTEN, I. H., NEAL, R. M., AND CLEARY, J. G. Arithmetic coding for data compression. *Communications of the ACM 30*, 6 (1987), 520–540.

[20] XIA, J. C., AND VARSHNEY, A. Dynamic view-dependent simplification for polygonal models. In *Visualization '96 Proceedings, IEEE* (1996), pp. 327–334.

RINT OF DRAWINGS
S ORIGINALLY FILED

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

1. Application papers are not suitable for scanning and are not in compliance with 37 CFR 1.52 because:
   - ☐ All sheets must be either A4 (21 cm x 29.7 cm) or 8-1/2"x 11". Pages _____ do not meet these requirements.
   - ☐ Papers are not ☐ flexible, ☐ strong, ☐ smooth, ☐ non-shiny, ☐ durable, and ☐ white.
   - ☐ Papers are not ☐ typewritten or mechanically printed ☐ in permanent ink ☐ on one side.
   - ☐ Papers contain improper margins. Each sheet must have a left margin of at least 2.5 cm (1") and top, bottom and right margins of at least 2.0 cm (3/4").
   - ☐ Papers contain hand lettering.

2. Drawings are not in compliance and were not scanned because:
   - ☐ The drawings or copy of drawings are not suitable for electronic reproduction.
   - ☐ All drawings sheets are not either A4 (21 cm x 29.7 cm) or 8-1/2" x 11".
   - ☐ Each sheet must include a top and left margin of at least 2.5 cm (1"), a right margin of at least 1.5 cm (9/16") and a bottom margin of at least 1.0 cm (3/8").

3. Page(s)_____ are not of sufficient ☐ clarity, ☐ contrast and ☐ quality for electronic reproduction.

4. Page(s)_____ are missing.

5. OTHER: NO Declaration

PTO/SB/68 (07-03)
Approved for use through 7/31/2003. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR ACCESS TO AN ABANDONED APPLICATION UNDER 37 CFR 1.14

RECEIVED
AUG 22 2003
File Information Unit

Bring completed form to:
File Information Unit
Crystal Plaza Three, Room 1D01
2021 South Clark Place
Arlington, VA
Telephone: (703) 308-2733

In re Application of	
Application Number	Filed
60/098150	8-26-99

Paper No. #2

I hereby request access under 37 CFR 1.14(a)(1)(iv) to the application file record of the above-identified ABANDONED application, which is identified in, or to which a benefit is claimed, in the following document (as shown in the attachment):

United States Patent Application Publication No. _____, page, _____ line _____,

United States Patent Number 6458266, column _____, line, _____ or

WIPO Pub. No._____, page _____, line _____.

Related Information about Access to Pending Applications (37 CFR 1.14):
Direct access to pending applications is not available to the public but **copies** may be available and may be purchased from the Office of Public Records upon payment of the appropriate fee (37 CFR 1.19(b)), as follows:
For **published** applications that are still pending, a member of the public may obtain a **copy** of:
the file contents;
the pending application as originally filed; or
any document in the file of the pending application.
For **unpublished** applications that are still pending:
(1) If the benefit of the pending application is claimed under 35 U.S.C. 119(e), 120, 121, or 365 in another application that has: (a) issued as a U.S. patent, or (b) published as a statutory invention registration, a U.S. patent application publication, or an international patent application publication in accordance with PCT Article 21(2), a member of the public may obtain a **copy** of:
the file contents;
the pending application as originally filed; or
any document in the file of the pending application.
(2) If the application is incorporated by reference or otherwise identified in a U.S. patent, a statutory invention registration, a U.S. patent application publication, or an international patent application publication in accordance with PCT Article 21(2), a member of the public may obtain a **copy** of:
the pending application as originally filed.

Robert W. Ackley
Signature

8-22-03
Date

Robert W. Ackley
Typed or printed name

Registration Number, if applicable

703553-0000
Telephone Number

RECEIVED
FOR PTO USE ONLY
Approved for File Information Unit
Unit: _____

This collection of information is required by 37 CFR 1.14. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **BRING TO: File Information Unit, Crystal Plaza Three, Room 1D01, 2021 South Clark Place, Arlington, VA.**

US006438266B1

(12) **United States Patent**
Bajaj et al.

(10) **Patent No.: US 6,438,266 B1**
(45) **Date of Patent: Aug. 20, 2002**

(54) **ENCODING IMAGES OF 3-D OBJECTS WITH IMPROVED RENDERING TIME AND TRANSMISSION PROCESSES**

(75) Inventors: **Chandrajit Bajaj**, Austin, TX (US); **Eric D. Petjan**, Watchung, NJ (US); **Valerio Pascucci**; **Guozhong Zhuang**, both of Austin, TX (US)

(73) Assignee: **Lucent Technologies Inc.**, Murray Hill, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/383,836**

(22) Filed: **Aug. 26, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/098,150, filed on Aug. 27, 1998.

(51) **Int. Cl.**[7] **G06K 9/36**; G06K 9/46

(52) **U.S. Cl.** **382/243**; 382/154

(58) **Field of Search** 382/154, 232, 382/240, 243; 345/419, 420, 423, 427; 375/240.08, 240.09

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,949,422 A * 9/1999 Mochizuki et al. 345/420
6,009,435 A * 12/1999 Taubin et al. 345/420
6,314,205 B1 * 11/2001 Masuda et al. 382/232

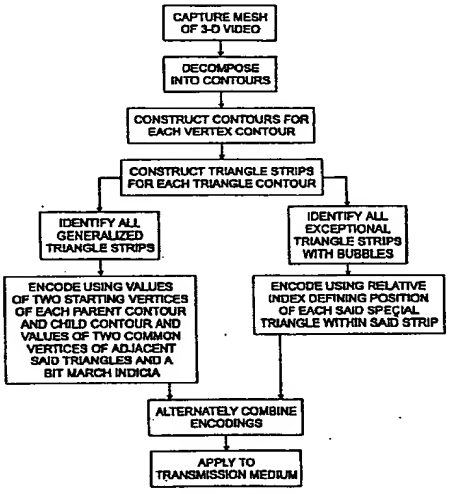
* cited by examiner

Primary Examiner—Timothy M. Johnson
(74) *Attorney, Agent, or Firm*—Charles E. Graves

(57) **ABSTRACT**

Disclosed is a scheme for creating, manipulating and transmitting images of a 3-D object through networks. A useful layering coupled with vector quantization allows an efficient combination of compressing and encoding of both the topology (connectivity) and geometry (vertex coordinates) of polygon meshes such as triangles. Connectivity information is contained in both generalized triangle strips and exception strips, the latter characterized by triangles for which all three vertices lie in the parent vertex layer. Geometry encoding is achieved with a productive vector quantization scheme which uses correction vectors for which distortion is merged into a following correction vector. Progressive connectivity transmission is achieved by using both intra-layer and inter-layer decomposition and reconstruction. The process encodes connectivity and geometry separately. Encoding of 3-D objects having left-right mirror symmetries using the process is described.

16 Claims, 30 Drawing Sheets



DECOMPOSITION AND ENCODING OF 3D IMAGE

POSITION		ID NO.	DATE
CLASSIFIER			
EXAMINER		7098	9-25-98
TYPIST			
VERIFIER			
CORPS CORR.			
SPEC. HAND			
FILE MAINT			
DRAFTING			

(LEFT INSIDE)

PATENT APPLICATION

60098150

APPROVED FOR LICENSE

INITIALS

Date Entered or Counted

CONTENTS

Date Received or Mailed

	1.	Application papers.	
	2.	Request for Access	8/28/12
	3.		
	4.		
	5.		
	6.		
	7.		
	8.		
	9.		
	10.		
	11.		
	12.		
	13.		
	14.		
	15.		
	16.		
	17.		
	18.		
	19.		
	20.		
	21.		
	22.		
	23.		
	24.		
	25.		
	26.		
	27.		
	28.		
	29.		
	30.		
	31.		
	32.		